AFRL-IF-RS-TR-1998-11
Final Technical Report
March 1998

LEARNING OBJECT AND SCENE RECOGNITION STRATEGIES

University of Massachusetts

Sponsored by
Advanced Research Projects Agency
ARPA Order No. AO-A869

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1998-11 has been reviewed and is approved for publication.

APPROVED: Peter J. Costianes

PETER J. COSTIANES
Project Engineer

FOR THE DIRECTOR:

JOSEPH CAMERA, Deputy Chief
Information & Intelligence Exploitation Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFEC, 32 Hangar Road, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

LEARNING OBJECT AND SCENE RECOGNITION STRATEGIES

Edward M. Riseman
Allen R. Hanson
Bruce Draper

Contractor: University of Massachusetts
Contract Number: F30602-94-C-0042
Effective Date of Contract: 30 December 1993
Contract Expiration Date: 30 December 1996
Program Code Number: 62301E
Short Title of Work: Learning Object and Scene Recognition Strategies
Period of Work Covered: Dec 93 - Dec 96

Principal Investigator: Bruce Draper
Phone: (413) 545-1320
AFRL Project Engineer: Peter J. Costianes
Phone: (315) 330-4030

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by Peter J. Costianes, AFRL/IFEC, 32 Hangar Road, Rome, NY 13441-4114.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1998	Final Dec 93 - Dec 96

4. TITLE AND SUBTITLE

LEARNING OBJECT AND SCENE RECOGNITION STRATEGIES

5. FUNDING NUMBERS

C - F30602-94-C-0042
PE - 62301E
PR - A869
TA - 00
WU - 01

6. AUTHOR(S)

Edward M. Riseman, Allen R. Hanson, and Bruce Draper

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Massachusetts
Computer Science Department
Box 34610 - Lederle GRC Bldg
Amherst MA 01003-4610

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Air Force Research Laboratory/IFEC
32 Hangar Road
Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-1998-11

11. SUPPLEMENTARY NOTES

Air Force Research Laboratory Project Engineer: Peter J. Costianes/IFEC/(315) 330-4030

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report summarizes activities performed as follows: (1) development of the Schema Learning System (SLS) from Draper thesis; (2) learning multi-variate decision trees for object classification; (3) real-time color classification in UGV domain via table look-up; (4) interactive real-time classification visualization for training data specification; (5) new formulation of approach to learning via Markov decision processes.

14. SUBJECT TERMS

Learning by Example, Object Recognition, Color Classification, Multi-Variate Decision Trees, Markov Decision Processes

15. NUMBER OF PAGES

80

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

0) Overview and Introduction

The final report will be a summary of activities as follows:

1) Schema Learning System (SLS) from Draper thesis as originally applied;

2) learning multivariate decision trees for object classification;

3) real-time color classification in UGV domain via table look-up

4) interactive real-time classification and visualization for training data specification

5) new formulation of approach to learning via Markov decision processes (led to new DARPA contract for Draper at Colorado State to continue this avenue of research).

Note that the last two sections above (#4 and #5) will serve as the last quarterly report of the contract.

In December of 1993, researchers at the University of Massachusetts began an ambitious project to try to build systems capable of learning object recognition strategies. Now, at the end of the project, we are pleased to look back at what we believe are significant accomplishments:

1) The Schema Learning System (SLS)

As originally proposed, we built the Schema Learning System (SLS), a system that learns recognition strategies by training a hierarchically-organized series of decision trees. The first version of this system was completed in late 1993, and experiments with it led to several publications [17,19,22]. Moreover, the first version of SLS was completed early enough in the project that we were able experiment with it and develop intuitions about its strengths and weaknesses, which in turn led to the development of a reinforcement learning version of the system (see point #5).

2) Learning multivariate decision trees for object classification

As a necessary part of building SLS, we became experts in applying decision tree technology to computer vision problems, and established a collaboration with the learning projects of Prof. Paul Utgoff. In the process, we discovered that linear machine decision tree (LMDT) classifiers perform surprisingly well at classifying pixels and regions in outdoor images. This lead to an initial key publication [18], and sparked an entire line of research probing the theoretical and practical foundation for this approach, as recently reported in [8].

3) Real-time color classification in the UGV domain via table look-up

The decision tree classification led to the development of an interesting application in the DARPA Unmanned Ground Vehicle (UGV) program for Automatic Target Detection (ATD), since it was amenable to rapid classification via table look-up methods. Understanding and exploiting this phenomenon led to another series of publications [8,9] and is being exploited both academically and at a major American automaker for obstacle detection and avoidance in automated highway applications.

4) Interactive classification and visualization for training data specification

The construction of pixel classifiers is a labor-intensive task involving user interaction to manually provide training data, select feature sets, and provide feedback as a result of classification. We have developed an exciting prototype interactive tool [52] that allows the user to immediately see the result of selecting incremental training data so that he can adjust the further selection on the basis of inaccurate classification. This is being developed into a modern 3D visualization tool for classification of terrain in aerial images, and will be included in an NSF proposal in the near future.

5) Learning object recognition strategies via Markov decision processes

Most importantly, experience with SLS convinced us that the level of control provided by decision trees was too coarse for the optimal control of computer vision algorithms. In particular, the sensitivity of the decision trees to the results of early stages of processing was too limited. This led to the development of a new technological approach -- a version of SLS based not on decision trees, but on reinforcement learning. This conceptual shift (reported in [17,20,21]) is a quite promising result, and has encouraged DARPA to invest in a new project to investigate the use of reinforcement learning to automatically populate geospatial databases (as part of the new DARPA APGD program).

I. The Schema Learning System (SLS)

I.1 Motivation

Over the last twenty years, the field of image understanding has divided into 10-20 (or more) subfields, each with a narrowly-defined problem focus. Within each subfield, theories have been developed and tested and different solution methodologies have been adopted. As a result, there are now several good and improving algorithms for edge and line extraction (straight and curved), feature tracking, depth from motion (two-frame and multi-frame), camera calibration and 3D pose determination, to name just a few of the areas in which progress has been made. Progress in 3D vision has been particularly strong; the advent of 3D IFSAR sensors and improvements in stereo processing now provide basic procedures for extracting and reasoning about 3D information. One of the areas in which relatively little progress has been made, however, is so-called "high-level" vision. We believe that the slow progress in many complex image understanding tasks results from the lack of a theory and methodology for control and integration in vision systems.

I.2 Approach of SLS

The goal of SLS is to learn optimal recognition strategies as a mechanism for constructing integrated vision systems from a library of vision algorithms. If this can be done with only limited human interaction -- i.e. image understanding algorithms and representations can be compiled into efficient executable systems that solve specific object recognition problems -- then a major break-through in the implementation of computer vision systems will have been achieved.

Our approach was to model the visual process as a sequence of representational transformations with interleaved binary decision problems. This approach begins with an image (possibly stereo or motion images), and then progresses through a series of intermediate representations (e.g. regions, line segments, surfaces, volumes) to the goal representation (e.g. the position and orientation of the target object. At each step in the

process, a binary decision process determines which entities (e.g. regions, lines, etc.) should be further processed and which should be rejected.

The control policy is built from a library of image understanding routines, some of which transform data from one representation to another, and some of which compute features describing existing representations. The learning algorithms proceed in two stages: in the first, it selects which transformation operators to use at each level of representation (as a function of features of the data instance). In SLS, we used explanation-based learning for this step. In the second stage, binary classifiers are trained for each level of representation to separate promising from suspect data instances. Neural networks, multivariate decision trees and instance-based classifiers were all tested for this purpose, but we generally had the most success using multivariate decision trees as the binary classifiers.

I.3 Representations and Formalisms

Formally, SLS learns recognition strategies that satisfy recognition goals. Recognition goals are provided by a teacher (i.e. the user), and specify a) the object to be recognized, b) the target representation, and c) accuracy thresholds. For example, in one of the experiments described below, the recognition goal was to recognize the (3D) position of the UMass engineering building. This was specified by giving a recognition goal to produce a 3D transformation (thus specifying the target representation), giving it a training signal composed of positions of the engineering building (relative to the camera), and providing a utility function that penalized for solutions that varied by more than 5% in the plane or 10% in scale.

To satisfy such recognition goals, SLS learns recognition strategies that control the application of visual procedures (VPs) to hypotheses. Visual procedures are algorithms from the computer vision literature, such as edge extraction, vanishing point analysis or model matching. VPs are thus analogous to knowledge sources in a blackboard system (e.g. [32,24]) or Ullman's visual routines [64]) in the sense that they are the procedural primitives used to build larger recognition strategies. Hypotheses are intermediate-level data items, such as edges or surfaces (or sets thereof). At each step in the recognition process, a VP is applied to one or more hypotheses to either 1) measure a feature of the hypothesis or 2) generate new, higher-level hypotheses. (Feature measurement procedures are referred to as FMPs, while transformational procedures are called TPs[3].)

Figure 1 shows the template for declaring a visual procedure. The goal is not to have the user "program" SLS through the VP library (in the sense that the knowledge engineer "programs" a blackboard system), but rather to have the system learn recognition strategies based on only syntactic information about the visual procedures. Therefore, the template contains only enough syntactic information about a VP to let SLS apply it to training images; any other information, such as the expected cost and/or quality of the VP, much be estimated by SLS. The VP template specifies how many hypotheses are required as (run-time) arguments, the level of representation of each argument, any prerequisite features, and a Lisp S-expression for invoking the VP. In addition, TP

---

[3] Although TPs are described as transformation procedures, the word `transformation' should not be construed as implying a one-to-one mapping between old and new hypotheses. TPs can combine information from multiple hypotheses (e.g. stereo) and may generate an arbitrary number of new hypotheses (e.g. segmentation). In addition, TPs do not consume their arguments, so multiple TPs may be applied to a single hypothesis. Some readers may therefore find it helpful to think of TPs as procedures that generate new hypotheses from old hypotheses, rather than as transformation operators.

VP Declaration Template

VP Name:	VP Name
Type:	Transformation of Feature Measurement
Arguments:	Number of run-time arguments (hypotheses)
Levels:	Level of representation for each argument
Prerequisites:	List of required hypothesis features
Result Level:	Level of representation of resulting hypothesis (TPs only)
Feature Value:	List of discrete feature values (FMPs only)
S-Expr:	Lisp s-expr to invoke procedure

Figure 1. VP Declaration Templates. Each VP declaration in the library includes enough syntactic information for SLS to apply the VP to training images. This includes the name of the VP, its type, the number of run-time arguments (hypotheses), the level of representation (and any prerequisites) of each argument, and either the discrete feature values (for a FMP) or the type of hypothesis generated (for a TP).

declarations include the type of hypothesis generated, while FMP declarations include the number of discrete feature values the FMP returns.

Recognition strategies are represented by recognition graphs, which are decision trees that have been generalized to multiple levels of representation. Borrowed from the field of operations research, decision trees are trees of alternating choice nodes and chance nodes designed to help managers make decisions about actions with uncertain outcomes [33]. Choice nodes in a decision tree represent decisions over which the agent (typically a business manager) has control, while chance nodes represent events the agent cannot control but whose likelihoods can be estimated. Using decision trees, managers estimate the probabilities of potential consequence of a decision or series of decisions before any action is taken. For example, a manager might consider investing in a new manufacturing facility. If the investment is made and the product sells there will be a profit, but there is some possibility that the product will not sell and the investment will be lost. This scenario can be represented by a decision tree with a choice node at the root representing the option to invest or not, and a chance node representing whether or not the product sells. In AI terminology, decision trees can be thought of as state-space representations similar to game trees with probabilistic opponents.

(Readers familiar with AI-style decision trees such as ID3 [55] will note that the choice nodes in such systems are omitted. These systems make all their choices while learning, leaving only the chances nodes in the tree. SLS does the same, pruning away every option but one at each choice node. Nonetheless, it will be convenient to leave the choice nodes in the formalism for when we describe the optimization algorithm that produces minimum-cost trees.)

The Schema Learning System (SLS) uses multi-level decision trees (called recognition graphs) in the belief that image data should not be matched directly to object models. Instead, a sequence of more and more abstract descriptions of the image data, represented as intermediate-level hypotheses, are built up under constraints provided by the object model, until goal-level hypotheses are eventually generated. Recognition can therefore be modeled as a sequence of representational transformations, beginning with an image and concluding with (an instance of) the target representation.

Fixed sequences of representational transformations tend to be brittle control strategies, however. Most visual procedures depend upon assumptions that are poorly understood and/or are "rules of thumb" that apply most of the time but not always. As a result, most procedures from the image understanding literature are subject to unexplained failures. It is therefore important to learn closed-loop control strategies that measure the quality of each intermediate representation as it is produced, and dynamically select and/or reject hypotheses. This is the role of the decision trees; the decision trees filter the good hypotheses from the bad ones at each level of representation, introducing a classification step between each representational transformation to provide the system with feedback.

Each level of a recognition graph is therefore a decision tree that measures features of a hypothesis, trying to judge whether it should be pursued further. Hypotheses that reach a positive leaf node of the decision tree are then used as arguments for transformational procedures that create new, more abstract hypotheses. (Hypotheses that reach a negative leaf node of a decision tree are discarded.) The overall recognition graph is therefore a stack of decision trees, one for each type of intermediate representation (see Figure 2). Actions within a decision tree are feature measurement procedures that collect information about a hypothesis in order to determine if it should be accepted or rejected. Hypotheses that reach positive terminal nodes are then transformed (via a VP) to a higher level of representation, where there enter another decision tree. The highest level of

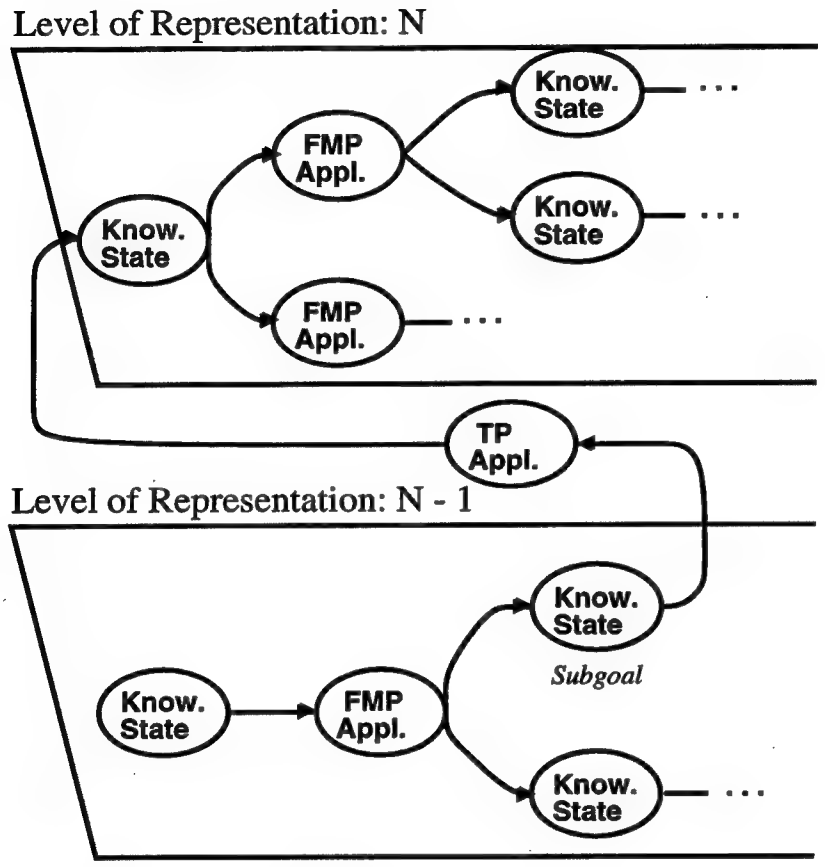


Figure 2. *A recognition graph. Levels of the graph are decision trees that verify hypotheses using feature measurement procedures (FMPs). Hypotheses that reach a subgoal are transformed to the next level of representation by transformation procedures (TPs).*

representation is the target representation (as specified by the recognition goal), and hypotheses that reach a positive leaf node of the top decision tree and the objects identified by the system.

I.4 Capabilities and Limitations of Recognition Graphs

So far, object recognition has been described as a "bottom-up" process starting with an image and ending with an abstract representation of an object. Although we will continue to use bottom-up terminology, it should be noted that recognition graphs can also represent "top-down" strategies and even mixed bottom-up and top-down strategies. "Bottom-up" strategies are created from TPs that create more abstract hypotheses from less abstract ones; top-down strategies are constructed from TPs that reduce abstract hypotheses to more concrete ones. Many strategies are mixed, using TPs that produce both more and less abstract hypotheses. The only constraint enforced by SLS on recognition graphs is that the VP library should not contain any loops, where hypotheses of type A are created from hypotheses of type B and vice-versa.

At the same time, recognition graphs are not capable of representing strategies based on relative strengths of hypotheses. Traditional blackboard systems can use heuristic schedulers that apply a knowledge source to the top N hypotheses at a level of representation, but such strategies cannot be embedded in recognition graphs. Recognition graphs can represent strategies that apply VPs to hypotheses with specific sets of features, but not to the N best hypotheses in an image. (This is why a minimum distance classifier is used to enforce the constraint that only one goal-level hypothesis by verified per image.)

SLS's strategies compare run-time hypotheses to training-time hypotheses. If training-time hypotheses with similar features led to correct goal-level hypotheses, then a hypothesis is pursued further; if not, it is rejected. SLS strategies base their control decisions not on the relative strengths of hypotheses from a single image, but on the relative strength of run-time hypotheses when compared to the larger (but less specific) pool of training hypotheses.

I.5 Algorithms

At the heart of SLS are algorithms that create recognition graphs from training images. SLS learns recognition graphs through a three step process of search, learning from examples, and graph optimization, as shown in Figure 3 The search algorithm looks for sequences of transformation procedures that produce correct goal-level hypotheses; in the process, it also estimates the costs and likelihoods associated with VPs and FMPs. The learning from examples algorithm inspects the operator sequences identified by the search algorithm and infers a generalized concept of how correct goal-level hypotheses are generated. Typically it will discover that in order to recognize an object reliably, several (possibly redundant) operators must be applied to certain types of hypotheses. Finally, the graph optimization algorithm creates decision trees at each level of the recognition graph that minimize the expected cost of verification. The result is a multi-level recognition graph representing an efficient and reliable strategy for identifying the target object in terms of the specified goal (e.g. 2D or 3D, approximate or exact).

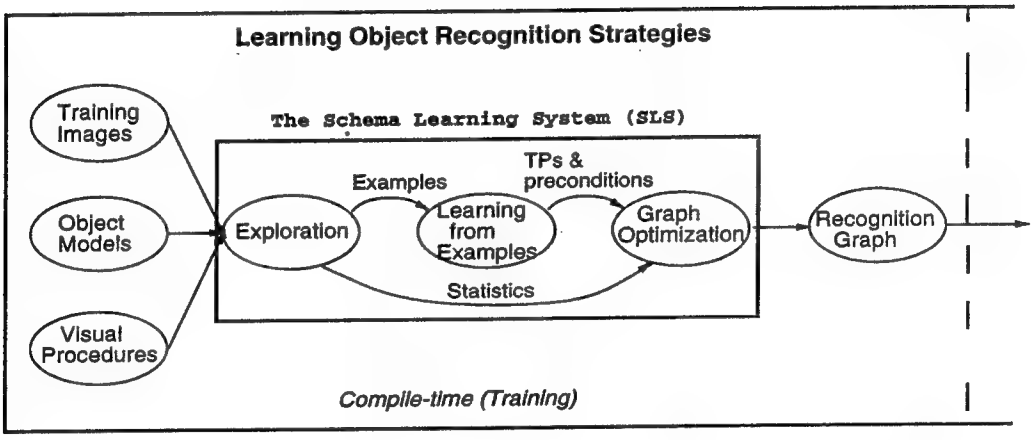


Figure 3. *The Three Algorithms of SLS. This figure expands the left-hand side of Figure 1 to show the search, learning from examples, and graph optimization modules that are the heart of SLS.*

I.5.1 Search

The search algorithm applies visual procedures to training images and to intermediate-level hypotheses generated from training images. It begins by applying TPs to the images, producing intermediate hypotheses such as regions, lines, and points. The properties of these hypotheses are measured by FMPs, and then they are transformed by TPs into still more abstract hypotheses. The search algorithm exhaustively expands the tree of hypotheses in this way until no new hypotheses can be generated.

There are two reasons for exhaustively searching the space of hypotheses that can be generated from training images. The first is to provide correct hypotheses for the LFE algorithm. The training signal provided by the user distinguishes correct goal-level hypotheses from incorrect ones, but it does not indicate how goal-level hypotheses can be generated from images through sequences of intermediate-level hypotheses. To learn to generate goal-level hypotheses, SLS needs examples of how correct hypotheses are created. By exhaustively generating all possible hypotheses, the search algorithm is guaranteed to create as many correct hypotheses as possible, and it saves a record of how each hypothesis was generated.

The second reason for exhaustively searching images is to estimate the costs and benefits of VPs. In order to optimize the verification process, SLS has to know the probability of a feature given a hypothesis, as well as the expected cost of measuring that feature. Unfortunately, SLS's VP library does not include any information about the costs of FMPs or the probabilities of each discrete feature value. SLS therefore has to build up a statistical characterization of the FMPs by applying them to training images.

Although SLS is designed to maximize run-time, rather than compile-time, efficiency, there are many situations where exhaustively expanding the tree of possible hypotheses is not feasible. In such cases, the cost of exploration can be heuristically reduced by not exploring hypotheses that do not satisfy spatial constraints derived from the training signal. For example, if the recognition goal is to recover the three dimensional position of an object, any region hypotheses that do not overlap the object's projection can be rejected without being explored further. Similarly, points, lines, planes, and other types of geometric hypotheses can be rejected if they fail to overlap the correct solution or its projection. In this way, the combinatoric nature of exploration is damped, but the positive examples required by the LFE algorithm are still generated.

The disadvantage of this heuristic is that negative examples are used in SLS 1) by the LFE algorithm, to select the minimal cost DNF subterm, and 2) to estimate the costs and probabilities associated with features. At the risk of a less efficient strategy, both tasks can be accomplished by exploring only a subset of negative hypotheses and extrapolating the results. We are currently experimenting with this and other heuristics for minimizing the search cost; nonetheless, the experiments in this paper were run using exhaustive search.

I.5.2 Learning from Examples (LFE)

SLS's learning from examples (LFE) algorithm analyses correct hypotheses produced during exploration and infers from them an efficient scheme for generating accurate goal-level hypotheses. The approach reflects the idea that recognition is a series of transformations interleaved with verifications. By looking at the histories of how correct

hypotheses develop, SLS learns how to generate goal-level hypotheses from images through series of intermediate-level hypotheses. At the same time, it learns which features of intermediate hypotheses indicate that a hypothesis should be pursued, and which imply that a hypothesis should be abandoned.

In the machine learning literature, the term learning from examples refers to algorithms that learn rules for evaluating examples. Following the terminology in the AI Handbook [12], learning from examples problems are defined in terms of instance spaces and rule spaces. The instance space is the set of possible examples or instances that might be encountered, either during training or testing. The rule space is the set of possible inference rules for evaluating instances. Learning from examples algorithms search rule spaces for the best methods of evaluating instances.

In SLS's LFE algorithm, the task is to generate correct goal-level hypotheses from images through sequences of intermediate representations. Instances are strings of hypotheses and TPs that lead from images to correct goal-level hypotheses. The rule space is composed of (sets of) features and TPs: the features determine which hypotheses should be pursued (at each level of representation), and the TPs indicate how they should be transformed. The goal of the LFE algorithm is to select a set of TPs and features that will generate a correct hypothesis for every target object instance in the training set, while generating as few false hypotheses as possible.

Inside the LFE algorithm, instances of correct hypotheses are represented as dependency trees. A dependency tree is an AND/OR tree recording the TPs and intermediate-level hypotheses on which a goal-level hypothesis depends. For example, a 3D pose hypothesis can be created by a geometric matching algorithm that finds the pose that minimizes the error of projecting a (3D) object model onto a set of (2D) image line segments. If so, the pose hypothesis is dependent on the geometric matching TP and the image line segments, as well as the TPs and hypotheses needed to generate the image line segments. In general, dependency is recursive, with 'AND' nodes resulting from TPs that require multiple arguments (and are therefore dependent on more than one hypothesis), and 'OR' nodes occurring when more than one TP redundantly generates the same hypothesis.

Dependency trees as described above apply to specific hypotheses generated during the search phase of SLS. The first step in inferring a more generalized scheme for generating goal-level hypotheses is to replace specific hypotheses with their feature vectors. The rationale for the substitution is that TPs have preconditions associated with them that select which hypotheses they should be applied to. If a TP needs to be applied to hypothesis H to ensure that a goal is met, then only features of H should be considered as preconditions for the TP.

In general, a hypothesis is guaranteed to be created by any set of preconditioned TPs that "satisfies" its dependency tree. A dependency tree DT is satisfied by a set of TPs G (with affiliated preconditions P) if: 1) the root of DT is an AND node, and every subtree of DT is satisfied; 2) the root of DT is an OR node, and at least one subtree of DT is satisfied; or 3) the root of DT is a leaf node with TP g and preconditions P such that g is in G and the preconditions of g either match or are a superset of P.

The algorithm for finding optimal sets of TPs and preconditions is deceptively simple:

- Convert the generalized dependency tree of a correct goal-level hypothesis to disjunctive normal form (DNF)[4]
- For every other correct goal-level hypothesis:
- Convert its generalized dependency tree to DNF.
- "AND" together the new DNF expression with the previous DNF expression.
- Convert the resulting 'AND' tree to DNF[5]
- Select the conjunctive subterm that generates the fewest total hypotheses.

By the logic of the dependency relation, the TPs and preconditions in any conjunctive subterm of the final DNF expression are sufficient to re-generate all correct goal-level hypotheses from the training images. By selecting the minimal term, SLS chooses the best method for generating correct hypotheses.

AND/OR dependency trees are converted to DNF by a standard algorithm that first converts every subtree to DNF and then either merges the subterms, if the root is an OR node, or takes the symbolic cross product[6] of the subterms, if the root is an AND node. If a TP is ANDed with itself when taking the cross product, the resulting preconditions are the intersection of the preconditions of the two instances being ANDed.

This basic algorithm is altered slightly to improve efficiency. Because SLS seeks to find the minimal term (measured as the number of hypotheses generated) of the DNF expression rather than every term, any conjunctive subterm that is a logical superset of another can be pruned, reducing the total number of terms considered. A second modification is to sort the correct goal-level hypotheses according to the size of their dependency trees and to iterate in step two from the simplest dependency trees to the most complicated. This reduces the size of the interim DNF expressions without affecting the final result.

I.5.3 Graph Optimization

As was stated earlier, recognition graphs interleave verification and transformation, using FMPs to measure properties of hypotheses and TPs to transform them to higher levels of representation. By building dependency trees from the training samples, converting them to DNF and picking the minimal subterm, SLS learns which TPs to use to transform hypotheses from one level to the next. Just as important, it learns which preconditions a hypothesis must meet before it should be transformed. These preconditions are the subgoals of the recognition process at intermediate levels of representation.

The optimization algorithm builds decision trees for each level of representation that minimize the expected cost of reaching a subgoal or, conversely, of deciding that a hypothesis cannot satisfy a subgoal and should be rejected. The decision trees are constructed by first building a graph representing all possible sequences of FMP applications, and then optimizing the graph by determining which options at each choice

---

[4] The disjunctive normal form of a logical expression is an OR of ANDs of monomial expressions, for example (**A** \wedge **B**) \vee (**A** \wedge **C**).}.

[5] Logically, this algorithm is equivalent to the simpler two-step process of ANDing all the dependency trees together and converting the result to DNF. However, iteratively adding each new dependency tree to an evolving expression simplifies the probabilistic analysis given in [23].}.

[6] Symbolic cross product: {A, B} \times {C, D} = {AC, AD, BC, BD}

node minimize the overall cost of recognition, and removing the other options. The final result is a decision tree at each level of representation that minimizes the expected cost of verification.

A preliminary step to building efficient decision trees is to characterize the performance of FMPs. In particular, SLS estimates:

- Expected Cost (VP, F), the expected cost of applying a VP to a hypothesis with the feature values F;
- Feature Likelihood (FMP, f1, F), the likelihood of a FMP returning feature value f1 when applied to a hypothesis with feature values F.

In general, these values are estimated from applications of FMPs to similar hypotheses during training. When an insufficient number of similar hypotheses (i.e. hypotheses with feature values F) are generated during training, the dependency on F is dropped and the values are estimated across all hypotheses.

Unfortunately, these statistical measures cannot be inferred directly from the search data, because the probabilities and costs associated with features depend on the quality of the hypotheses being measured. The search algorithm, which exhaustively explores the space of possible hypotheses, generates more hypotheses of lesser quality than SLS's run-time recognition strategy will. The exploration hypotheses are therefore drawn from a different statistical distribution than the run-time hypotheses will be.

As a result, the estimations of FMP performance are delayed until after the LFE algorithm has been run. The results of LFE are used to prune the exploration data, removing those hypotheses that are merely artifacts of exhaustive search and would not be generated using the VPs and preconditions selected by the LFE algorithm. Once the search data has been pruned, the remaining hypotheses are used to characterize the performance of VPs.

For each level of representation, a directed acyclic graph is constructed representing all possible sequences of FMP applications. The graph starts from a single knowledge state, corresponding to a newly generated hypothesis for which no features have been computed. The start state, like all knowledge states, is a choice node, since the control program gets to choose which FMP to apply first. FMP applications nodes are therefore added for every feature that can be measured of a hypothesis in the start state. These FMP application nodes lead to new knowledge states (one for each possible feature value), which in turn have more FMP applications attached to them, and so on. The expansion of the graph continues until it reaches either a subgoal knowledge state or a knowledge state that is incompatible with every remaining subgoal (i.e. a failure state).

More formally, we refer to subgoal states and failure states as the terminal states for each level of the recognition graph. The cost of promoting a hypothesis from knowledge state n to a terminal state is called the Expected Decision Cost (EDC) of knowledge state n, and the expected cost of reaching a terminal state from state n using FMP v[7] is the Expected Path Cost (EPC) of n and v. Since features are discrete, we denote the possible outcomes of a FMP v as a set R(v), and the probability of a particular feature value f being returned as $P(f \mid v,n)$, $f \in R(v)$.

---

[7] v is an awkward abbreviation for a feature measurement procedure, but **f** will be used for feature values and **p** would look like a probability value. Since FMPs are a subclass of VPs, **v** is therefore used.

The EDC's of knowledge states can be calculated starting from the terminal states and working backward through the recognition graph. Clearly, the EDC of a subgoal or failure state is zero:

$$EDC(n) = 0, \quad n \in \{\text{terminal states}\}.$$

The expected path cost of reaching a terminal state from a FMP application node is:

$$EPC(n,v) = C(v) + \Sigma_{f \in R(v)} (P(f \mid n,v) * EDC(n \cup f))$$

where n is a knowledge state expressed as a set of feature values, $n \cup f$ is the knowledge state that results from FMP v returning feature value f, and C(v) is the estimated cost of applying v.

The EDC of a knowledge state, then, is the smallest EPC of the FMPs that can be executed from that state:

$$EDC(n) = \min_{v \in VP(n)} (EPC(n, v))$$

where VP(n) is the set of FMPs applicable at node n. The minimal-cost decision tree is created by making a single pass through the directed acyclic graph, starting at the terminal nodes and working backward toward the start state. At each knowledge state, the pruning process calculates the EPC of every FMP that can be applied from that state, and removes all FMP application nodes except the one with the smallest EPC. The final result is a minimal-cost decision tree.

The equations above establish a mutually recursive definition of the expected decision cost of a knowledge state. The EDC of a knowledge state is the EPC of the optimal FMP application from the state; the EPC of a FMP application is the expected cost of applying the FMP plus the expected EDC remaining after the FMP has been applied. The recursion bottoms out at terminal nodes, whose EDC is zero. Since every path through the object recognition graph ends at either a subgoal or a failure node, the recursion is well defined.

Furthermore, the total cost of recognition can be estimated from the EDCs of start states and the expected costs of the TPs selected by the LFE algorithm. The EDC of the start state for a level of representation estimates the expected cost of verifying or rejecting hypotheses at that level. By estimating the total number of hypotheses generated at each level by the preconditioned TPs and multiplying it by the EDCs of the start states, the total cost of verification can be estimated. Since the expected number of times a TP will be executed can also be estimated from the LFE algorithm's results, the total expected cost of recognition can be obtained easily.

I.6 Applications of SLS

I.6.1 Overview

The SLS system described above was tested on the task of learning to recognize the position of the UMass engineering building, in images taken from a pathway on campus. In the second, the goal was to recognize the position of the Lederle GRC (a more complex building) in images taken from arbitrary ground positions. The first experiments were quite successful. Using a "leave one out" methodology, we would train the system

on nineteen training images and apply it to a twentieth image that was not part of the training set; in nineteen out of twenty experiments, the control strategy learned by SLS found the building, and in sixteen of twenty it returned a pose that was correct to within 10% depth and 5% of all other external viewing parameters.

I.6.2 Training Data

The training data is selected from a set of twenty-one images collected along a hundred foot stretch of a footpath on the UMass campus. Figures 4a and 4b show the first and last images of the sequence. The images were taken level to gravity (±1°) and from approximately four feet above the ground, although the ground rises and falls over the course of the sequence. The camera was also subjected to small rotations in pan from one image to the next. As a result, the pose of the camera has four degrees of freedom, with large variations in position in the ground plane and smaller deviations in camera height and pan.

The "ground truth" positions and orientations of the Marcus Engineering building were determined by manually matching image points to model points and applying Kumar and Hanson's algorithm [38] to determine the building's pose relative to the camera. The training signal is therefore composed of errorful pose estimates, rather than true positions. However, Kumar and Hanson's results suggest that, with correct correspondences, their algorithm produces pose estimates that are extremely accurate when compared to the relatively lax error thresholds in the recognition goal. The estimated poses can therefore reasonably be used as a training signal.

I.6.3 The Recognition Goal

The recognition goal is to find the pose of Marcus Engineering relative to the camera. Pose hypotheses are represented as rotation matrices with translation vectors, in the traditional

$$P' = R\,P + T$$

representation, where R and T are the rotation matrix and translation vector that transform a set of points P in the model's coordinate system into a set of points P' in the camera's coordinate system. Unfortunately, errors expressed in terms of R and T tend to be unintuitive, since if an object is rotated slightly about its center, this will be represented as a rotation about the focal point, counteracted by a large translation[8]. It is helpful, therefore, to express the error tolerances in a different representation.

Since the pose of the building has only four degrees of freedom, the tolerance thresholds in the recognition goal are expressed in terms of scale, image position, and object angle. These parameters reflect the fact that the pose of the building can be expressed as a vector from the focal point to any known point on the building, plus a horizontal rotation about the known point (remember that the building has no tilt or roll relative to the camera). Errors in the positional vector are expressed as an error in length, measured as a percent of the true camera-to-object distance, and an error in position, measured as an angle. (Since we are interested in the magnitude of the orientation error, not its direction, this

[8] The size of the counteracting translation is a function of both the extent of the rotation and the distance from the object center to the focal point.

can be written as a scalar.) Errors in the rotation of the object are also represented as an angle, this time about the axis of gravity.

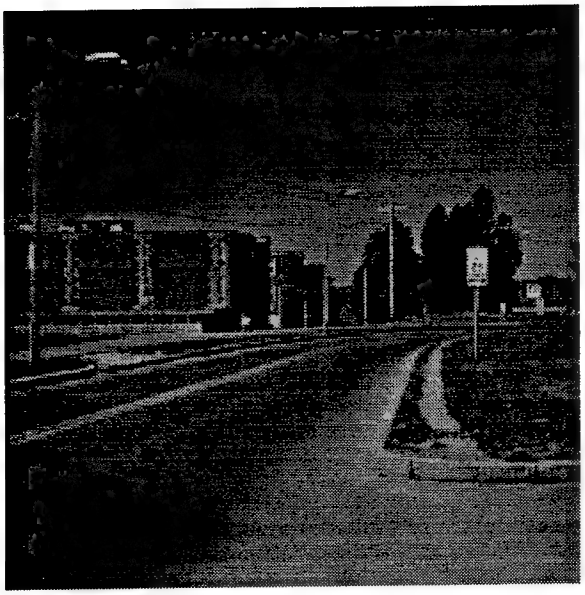


Figure 4a. *The first of twenty-one training images. The images were taken along a hundred-foot section of the path, with the camera level to gravity.*

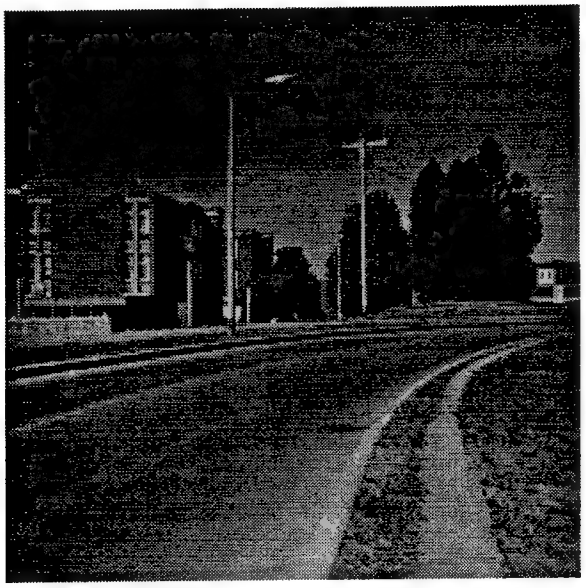


Figure 4b. *The last of twenty-one training images. The pose of the camera has four degrees of freedom, with large variations in position in the XZ (ground) plane and small differences in camera height and pan.*

The error thresholds for this exercise require the position of the building to be correct to within one degree of image angle and ten percent depth, while the orientation of the building must be correct to within five degrees around the axis of gravity. This implies that the hypothesized building poses should be highly accurate with respect to image position and reasonably accurate in orientation, but only approximate in depth. Figure 5 shows an example of a pose that satisfies these criteria, in this case the building pose identified by SLS's strategy in the first of twenty-one trials.

I.6.4 The Visual Procedure Library

The visual procedure library used in this experiment is shown in Table 6. It includes many procedures for extracting and grouping two-dimensional representations such as points and lines. Lines can be extracted using the edge-linking algorithm of Boldt and Weiss [4], and regions can be extracted by the algorithm described in Beveridge, et. al. [2]. Regions can also be created by fitting a convex hull to a set of line segments. Regions that match an expected color and texture can be selected from a region segmentation by a multivariate decision tree [6,18]. (This algorithm is included twice in the library with two different parameterizations, one designed to select red brickface regions, the other highly textured window regions.) Nearby regions can be grouped by a region merging TP, while another TP groups lines that intersect a given region. Nearby lines that are parallel, collinear or orthogonal can be grouped according to the relations defined by Reynolds and Beveridge [56]. (All of the grouping VPs are implemented using the facilities of the ISR database system [7].) Image points are extracted by finding trihedral junctions of lines.

The procedure library also includes routines that create or consume three-dimensional representations. Orientation hypotheses represent the orientation, but not location, of a plane in space, while planar surface hypotheses specify both the orientation and location of a plane. Most importantly, transformation hypotheses represent a coordinate transformation from one coordinate system to another, represented as a rotation matrix and a translation vector. Transformation hypotheses determine the pose of a modeled object by giving the transformation from the object model coordinate system to the camera coordinate system, and are goal-level hypotheses in this demonstration.

Three dimensional hypotheses are generated and manipulated by many visual procedures. Collins and Weiss [14] provide an efficient TP for grouping line segments into pencils, which are sets of lines that meet at a common point of intersection. Vanishing point analysis [14] infers the orientations of planes in space by assuming that the image lines in a pencil are the projections of parallel lines in space. Another approach to inferring the orientation of an object in space is to find trihedral junctions of line segments first, and then use the perspective angle equations of Kanatani [36] to infer the orientations of the planes, assuming the lines form right angles, like the corner of a building.

The distance from an object to the camera can be estimated when the size of the object is known; in the case of Marcus Engineering, the size of the building (and its wire-frame model) was extracted from its blueprints. Two parameterizations of the scaling TP are available in the VP library, one that estimates distance based on the apparent width of a window and the estimated angle of the building face, and a second that estimates distance from the height of the building using a direct inverse relationship of size to distance. (Note that since the images have zero tilt, the orientation of the building face is not needed to estimate distance from the building's height). Of course, since any two points on the object model can serve as compile-time parameters to a scaling TP, many other parameterizations of the scaling TP could be included in the library.

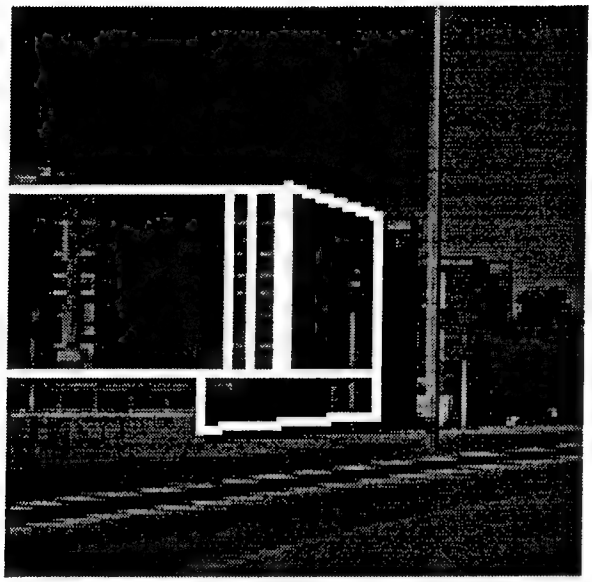


Figure 5. *A correct pose from one trial of the Marcus recognition strategy. The pose shown here was generated and verified on the first of twenty-one trials, and is off by 1.8% in depth, 4.79 degrees in orientation about the axis of gravity, and 0.16 degrees in image location.*

I.6.5 Results

Table 1 summarizes the results of twenty-one trials of learning to recognize the pose of Marcus Engineering from an approximately known viewpoint. The right side of the table shows the errors in the best goal-level hypothesis generated, even if this hypothesis was never verified, while the right side shows the errors in the goal-level hypothesis verified by the minimum distance classifier. The verified pose for trial number one, which is also the best pose generated for that trial, was shown earlier in Figure 5.

Pose errors in Table 1 are measured in terms of the length and orientation of a vector from the focal point to the corner of the building, and the rotation of the building. More precisely, the error in the position of the building is measured as 1) the error in the distance to the building, measured as a percentage of the true distance, and 2) the image position of the building, measured by the angle between the true vector from the focal point to the building corner and the estimated vector (labeled "Im Pos" in Table 1). The error in the building's orientation is measured as the angle about the gravitational axis between the estimated orientation of a building face and its true orientation (labeled "Rot." in Table 1).

The most striking feature of Table 1 is the result of trial sixteen. The strategy learned by SLS in trial sixteen did not generate a single goal-level hypothesis, either correct or incorrect, for the test image. An a posteriori analysis reveals that in twenty of the twenty-one images, the corner of the building is marked by a trihedral junction of image lines. In one image, however, noise eliminates one of the three lines. As a result, when the image without the trihedral junction is removed from the training set (and used as the test image), SLS learns a strategy that relies entirely on finding trihedral junctions. The strategy does not succeed in finding any trihedral junctions in the test image, however, and therefore generates no goal-level hypotheses. Ironically, in the other twenty trials, the training sets include the case in which trihedral junctions fail, and therefore the other twenty strategies all include redundancy to account for the possibility of trihedral failure, a redundancy that is never needed for the test images to which they are applied.

Trial sixteen is the only case in which the strategy learned by SLS fails to generate a correct goal-level hypothesis, giving it a success rate at generating 3D building hypotheses of 95%. In trials fifteen and twenty, however, SLS verified the wrong hypothesis, giving it an over-all success rate of 86%.

Table 1 addresses the robustness of the strategies learned by SLS, but not their efficiency. Table 2 shows SLS's expected run-time (in seconds, rounded to the nearest whole second) for the strategy learned in each trial as compared to the actual run-time when the strategy was applied to a test image. On any given trial, the discrepancy between the predicted and actual run-times is quite large. On average, however, the predicted run-times are within five percent of the actual run-times. This reflects the average-case nature of expected costs. The actual cost of recognizing an object in an image depends critically on the contents of the image, but as long as the training images are indicative of the test domain the average cost of recognition can be estimated. Indeed, an a posteriori analysis of the data shows that the five percent overestimate of the expected run time was caused by VPs executing more quickly during testing than during exploration, due primarily to variations in paging.

Table 1. Results of twenty-one trials of learning to recognize the pose of the Marcus Engineering Building. The left side of the table shows the errors in the best goal-level hypothesis generated for each trial, while the left side shows the errors in the hypothesis verified by the minimum distance classifier. Errors are specified in terms of the parameters discussed in Section 6.4.2, namely: 1) error in distance from the object to the camera, expressed as a percentage of the true distance from the object to the camera; 2) error in image position, measured in degrees; and 3) error in the building's orientation, measured in degrees.

	Best Generated Pose			Selected Pose		
Trial	Dist.	Rot.	Im Pos	Dist.	Rot.	Im Pos
1	1.81	4.79	0.16	1.81	4.79	0.16
2	1.34	0.82	0.05	1.34	0.82	0.05
3	1.18	1.33	0.11	2.74	1.33	0.09
4	0.69	1.40	0.13	1.97	1.40	0.13
5	2.64	2.33	0.10	2.64	2.33	0.10
6	0.73	6.95	0.15	0.73	6.95	0.15
7	0.27	1.16	0.05	6.58	1.16	0.07
8	2.33	0.07	0.08	2.33	0.07	0.08
9	1.01	3.58	0.20	1.69	3.58	0.20
10	1.39	1.65	0.04	2.07	1.65	0.05
11	0.50	3.27	0.08	2.37	3.27	0.25
12	2.24	4.48	0.25	2.24	4.48	0.25
13	2.07	1.54	0.04	2.07	1.54	0.04
14	0.36	1.63	0.09	0.36	1.63	0.09
15	2.13	2.58	0.21	11.23	2.58	0.21
16	–	–	–	–	–	–
17	4.44	6.21	0.13	4.44	6.21	0.13
18	1.07	1.75	0.09	1.77	1.76	0.16
19	0.41	3.83	0.07	1.07	3.83	0.07
20	0.92	2.50	0.03	14.64	2.50	0.03
21	1.18	4.95	0.13	2.26	4.95	0.13

Table 2. Timing results for the twenty-one Marcus Engineering trials. The first row shows the expected cost (in seconds, rounded to the nearest whole second) of applying the strategy, as predicted by SLS. The second row shows the actual cost.

Trial	1	2	3	4	5	6	7	8	9	10	11
Exp.	82.9	84.7	84.2	78.6	85.1	85.0	84.9	85.0	85.2	85.3	84.6
Act.	107.3	88.4	79.4	113.7	88.9	74.8	66.2	71.7	72.6	67.4	74.5

Trial	12	13	14	15	16	17	18	19	20	21	Avg.
Exp.	86.2	85.1	85.4	83.2	61.3	79.6	85.5	84.9	80.1	85.7	83.0
Act.	61.4	89.4	73.2	82.7	45.5	62.3	74.4	69.2	76.6	57.4	79.3

II. Learning multivariate decision trees for color object classification

Another branch of research pursued under this contract involves training multivariate decision trees (MDTs) to compensate for natural chromatic variation in outdoor images. We started looking at multivariate decision trees to verify hypotheses inside SLS control strategies. Our research expanded to using MDTs to classify color data, however, in part because color features are not useful in outdoor imagery otherwise, and in part because MDTs seem to perform surprisingly well at this task. Our goals were therefore twofold: we wanted to learn control strategies that combine color-based information with shape and other features even in outdoor domains, and we wanted to understand why MDTs learn to compensate for color variation so well.

II.1 Defining the Outdoor Classification Problem

The color (or rather, the apparent color) of an object depends on the illuminant color, the reflectance of the object, illumination geometry (orientation of the surface normal with respect to the illuminant), viewing geometry (orientation of the surface normal with respect to the sensor), and sensor parameters [34]. In outdoor images, the color of the illuminant (i.e., daylight) varies with the time-of-day, cloud cover and other atmospheric conditions [35]; the illuminant and viewing geometry vary with changes in object and camera position and orientation. In addition, shadows and inter-reflectances [33], and certain sensor response parameters [49], all of which can be difficult to model in outdoor scenarios, may also affect the apparent color of objects. Consequently, at different times of the day, under different weather conditions, and at various positions and orientations of the object and camera, the apparent color of an object can be different.

Figure 6 shows the variation in the apparent color of two simple matte surfaces (white and green) under different lighting and viewing conditions from about 50 images; the Figure also shows the color of each surface from one sample (represented by a single point), and the overall distribution in RGB space over the 50 images. In this example, the overall variation for each surface is about 250% of the distance between the centroids of the two clusters. In other words, the variation in the apparent color of a single surface can be greater than the difference (in color-space distance) between two distinct colors (white and green, in this case). The variation in the apparent color of more realistic objects, such as a road surface and a camouflaged military vehicle (Figure 7), can be even greater.

Human beings have an adaptive mechanism called color constancy that compensates for this color shift. Unfortunately, no corresponding adaptive mechanism exists in machine vision systems, and the notion of a color associated with an object is precise only within the context of scene conditions. Previous approaches have attempted to recognize object color without context or sufficiently robust models, and consequently have produced methods for color recognition that are effective only in highly constrained imagery.

In this project we analyzed variations in the apparent color of objects with respect to existing models of daylight and surface reflectance, and showed that the shift in apparent color under outdoor conditions can be represented by characteristic distributions in RGB space. We then showed that such distributions can be "learned" from training samples using Multivariate Decision Trees (MDT's) [6] for non-parametric approximation of decision boundaries around the training samples. Image pixels are then classified according to their location with respect to the learned decision boundaries.

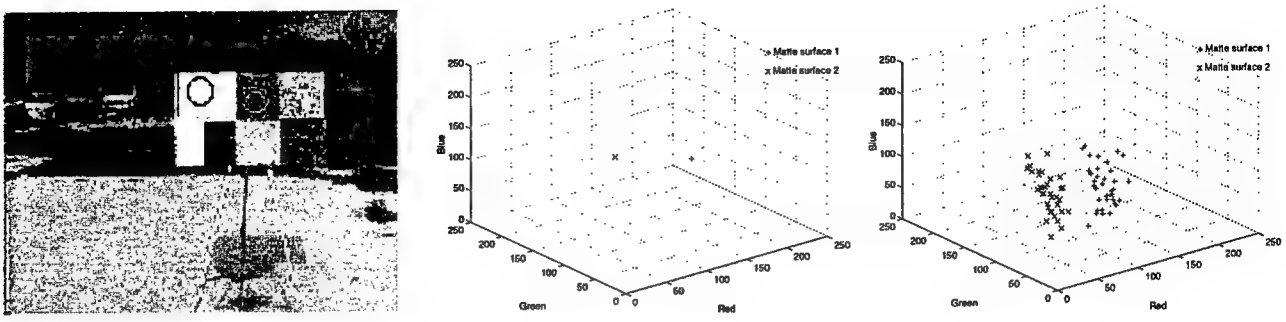


Figure 6. Variation of apparent color in outdoor images: (left to right) samples from two matte surfaces (extracted from circles), the *RGB* color from a single image, and the variation over 50 images.

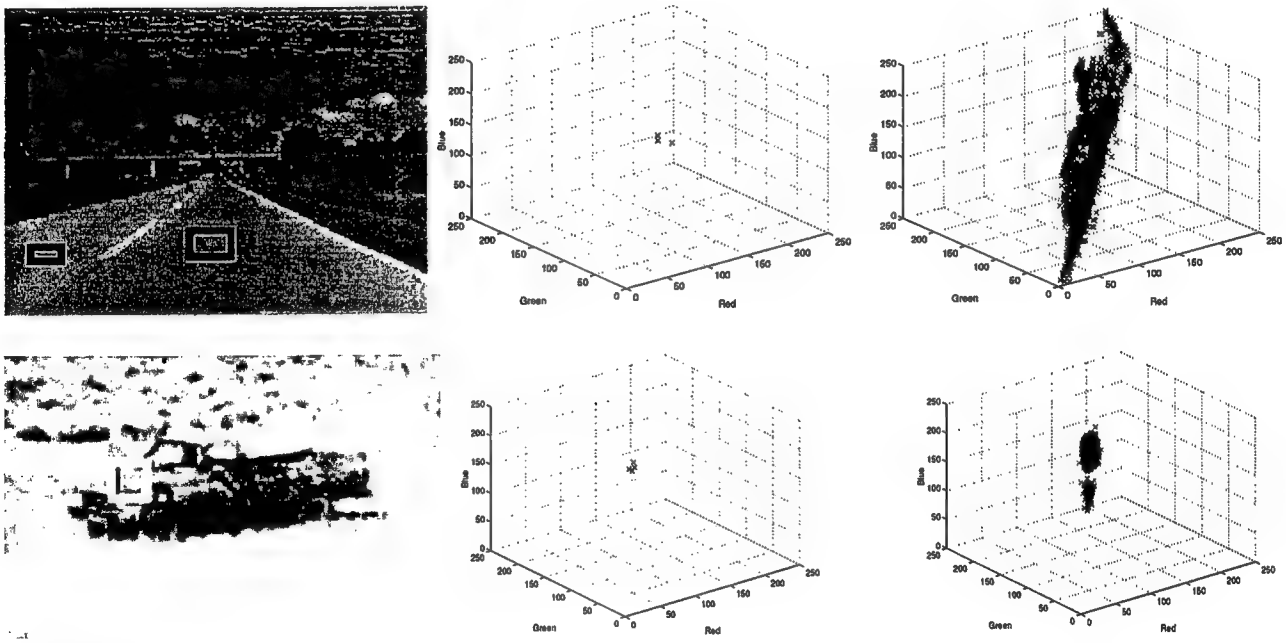


Figure 7. Samples of objects in real outdoor applications - highway road surface (top) and camouflaged military vehicle (bottom), the (RGB), color from a single image (sample color extracted from iation over about 100 images.

II.2 Previous Work in Outdoor Color Classification

Past work in color-based recognition under varying illumination can be divided into two categories: computational color constancy and non-parametric (sample-based) classification.

II.2.1 Previous Work on Color Constancy

Most of the work in computational color recognition under varying illumination has been in the area of color constancy, the goal of which is to match object colors under varying illumination without knowing the spectral composition of either the incident light or the surface reflectance; the general approach is to recover an illuminant-invariant measure of surface reflectance by first determining the properties of the illuminant.

Depending on their assumptions and techniques, color constancy algorithms can be divided into the following six categories [30]: (1) those which make assumptions about the statistical distribution of surface colors in the scene, (2) those which make assumptions about the types of reflection and illumination, (3) those assuming a fixed image gamut, (4) those which obtain an indirect measure of the illuminant, (5) those which require multiple illuminants, and finally, (6) those which require the presence of surfaces of known reflectance in the scene.

Among the algorithms that make assumptions about the statistical distributions of surface colors in the scene, Buchsbaum [11] assumes that the average of the surface reflectances over the entire scene is gray (the gray-world assumption); Gershon [31] assumes that the average scene reflectance matches that of another known color; Vrhel [67] assumes knowledge of the general covariance structure of the illuminant, given a small set of illuminants, and Freeman [29] assumes that the illumination and reflection in a scene follow known probability distributions. These methods are effective when the distribution of colors within the scene follows the assumed model or distribution. In outdoor scenes, the CIE daylight model [35] suggests that the gray-world assumption will not be valid; at the same time, as later sections will show, no general assumptions can be made about the distribution of surface colors even if the distribution of daylight color is known. Consequently, these methods are too restrictive for all but very constrained scenes.

The second set of color constancy algorithms make assumptions about the dimensionality of spectral basis functions [62] required to accurately model illumination and surface reflectance. For instance, Maloney [44] and Yuille [68] assume that the linear combination of two basis functions is sufficient. Under the assumption, the variation in surface color in a three-dimensional color space would follow a plane. Daylight, however, follows a parabolic surface in three dimensions RGB [8]; hence, the assumptions of these methods are true only under specifically controlled illumination.

Among the algorithms that make assumptions about image gamuts is Forsyth's CRULE (coefficient rule) algorithm [28], which maps the gamut of possible image colors to another gamut of colors that is known a-priori, so that the number of possible mappings restricts the set of possible illuminants. In a variation of this algorithm, Finlayson [27] applies a spectral sharpening transform to the sensory data in order to relax the gamut constraints. The assumptions about gamut-mapping restrict the application of CRULE to matte Mondrian surfaces under controlled illumination and fixed orientation. Ohta [50] assumes a known gamut of illuminants (controlled indoor lighting that lies on some

points along the CIE model), and uses multi-image correspondence to determine the specific illuminant from the known set. By restricting the illumination, this method is applied only to synthetic or highly constrained indoor images.

Another class of algorithms uses an indirect measure of the illumination. For instance, Shafer [61], Klinker [37] and Lee [43] use surface specularities (Sato [58] uses a similar principle, but not for color constancy); similarly, Funt [30] uses inter-reflections to measure the illuminant. These methods are based on the assumption of a single point-source illuminant; this assumption is not valid for an extended or non-point-source illuminant such as daylight.

In yet another approach, D'Zmura [26] and Finlayson [27] require light from multiple illuminants incident upon the multiple instances of a single surface in the same scene. The problem with these approaches is that they require identification of the same surface in two spatially distinct parts of the image that are subject to different illuminants. Once again, the approaches have been shown to be effective only on Mondrian or similarly restricted images.

The final group of color constancy algorithms assumes the presence of surfaces of known reflectance in the scene and then determine the illuminant. For instance, Land's Retinex algorithm [42] and its many variations require the presence of a surface of maximal (white) reflectance within the scene. Similarly, Novak's supervised color constancy algorithm [49] requires surfaces of other known reflectances. Such assumptions, while applicable to controlled settings, are not generally applicable to unconstrained images.

The assumptions made by the aforementioned algorithms are such that most of them perform only on highly restricted images (such as Mondrians), under mostly constrained lighting. Forsyth [28] aptly states, "Experimental results for [color constancy] algorithms running on real images are not easily found in the literature. Some work exists on the processes which can contribute to real world lightness constancy, but very little progress has been made in this area."

II.2.2 Previous Work on Non-parametric Color Classification

The emergence of road-following as a machine vision application has spawned several methods that use color for road-following without specific parametric models. The SCARF algorithm [15] approximates an "average" road color from samples, and models the variation of the color of the road under daylight as Gaussian noise about an empirically derived "average" road color; pixels are then classified based on minimum-distance likelihood. This technique was successfully applied to road-following, but cannot be applied for general color-based recognition of road-scene objects. For instance, in the case of the examples in Figures 6 and 7, this approach would calculate an average color for each surface from the corresponding distribution, and use that average as the most likely color of the object under any set of conditions.

Pomerleau's ALVINN road-follower [54] uses color images of road scenes along with user-induced steering signals to train a neural network to follow road/lane markers. Although the ALVINN algorithm made no attempt to explicitly recognize lanes or roads, it showed for the first time, that a complex visual domain with unmodeled variation can be approached as a non-parametric learning problem. This approach represents a significant advance in road-following methodology; however, it is designed specifically

for road-following and is hence not applicable to color-based recognition of road-scene objects.

II.3 Color shift in outdoor scenes: Causes and analysis

The standard model of image formation [34] describes the observed color of objects in an image as a function of (i) the color of the incident light (daylight, in the case of outdoor images), (ii) the reflectance properties of the surface of the object (iii) the illumination geometry, (iv) the viewing geometry, and (v) the imaging parameters. Theoretical parametric models exist for all the phases of this process. Unfortunately, these models have not been proven effective in unconstrained color imagery for model-based color recognition; still, they provide an approximate qualitative description of the variation of apparent color. Figure 8 shows a pictorial description of the various processes involved in the formation of outdoor images. The pertinent models are described below, and a general hypothesis about RGB distributions representing apparent object under daylight is developed thereafter.

II.3.1 Illumination

Daylight is a combination of sunlight and (ambient) skylight; the variation in the color of daylight is caused by changes in the sun-angle, cloud cover and other weather conditions. The CIE daylight model [35] describes the variation in daylight color as a parabola in the CIE chromaticity space. (Figure 9).

$$y = 2.87\,x - 3.0\,x^2 - 0.275,$$

where $0.25 <= x <= 0.38$. In RGB space, the parabola stretches out into a thin paraboloid surface [8].

II.3.2 Illumination geometry and viewing geometry

Illumination geometry, i.e., the orientation of the surface normal with respect to the illuminant, affects the composition of the light incident upon the surface. The surface orientation determines how much light from each of the two components of daylight (sun and sky), is incident upon the surface. For instance, a surface that faces the sun is illuminated mostly by sunlight, whereas surfaces that face away are illuminated by the ambient light. The viewing geometry, which is the orientation of the surface with respect to the camera, determines the composition and magnitude of the light reflected onto the camera; this is primarily a function of the reflectance properties of the surface. For instance, a matte (Lambertian) surface has uniform reflection in all directions, whereas a shiny (specular) surface reflects light only along that angle of reflection which equals the angle of incidence.

II.3.3 Surface reflectance

The effect of illumination geometry and viewing geometry depends on the reflectance properties -- upon the strength of the specular component, to be precise. Most realistic surfaces have components of both Lambertian and specular reflection. A number of models have been applied with varying degrees of success to such surfaces, most notably

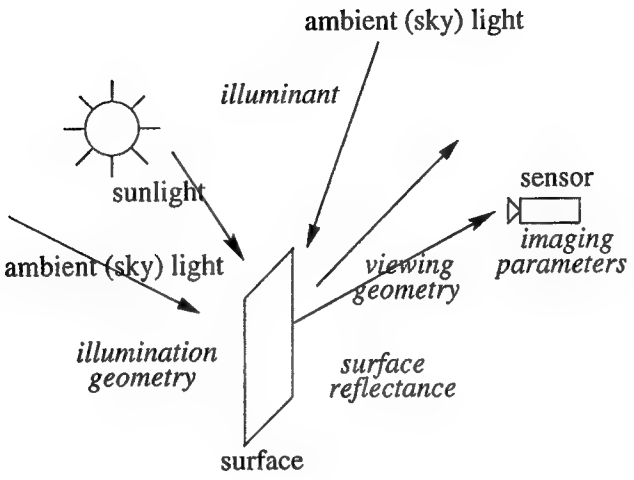


Figure 8. Image formation in outdoor scenes, along with the various processes involved.

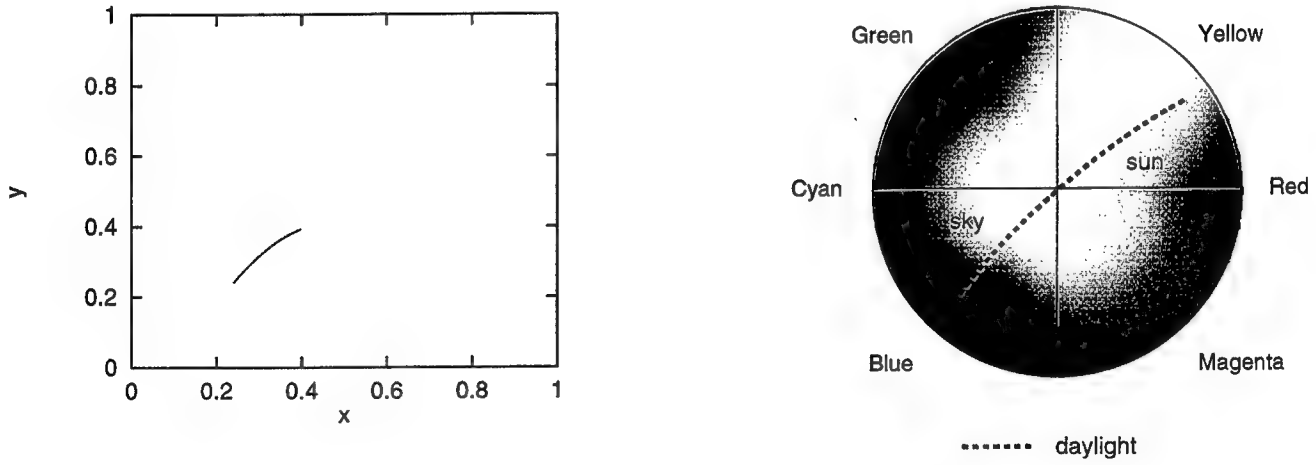


Figure 9. The CIE parametric model of daylight in the chromaticity space (left) and the color circle (right). The regions of the color circle representing the colros of sunlight and skylight (empirically determined) are shown.

Phong's shading model [51], Shafer's Dichromatic model [61], and Nayar's hybrid reflection model based on photometric sampling [46].

The Dichromatic reflection model (originally proposed by Shafer [61] and subsequently extended by Novak [48], Lee [43] and Klinker [37]) models the net reflection of a surface as the linear combination of the specular and Lambertian reflection components.

$$L(\lambda,i,e,g) = m_S(i,e,g)\, c_{S(\lambda)} + m_b(i,e,g)\, c_b(\lambda)$$

where $L(\lambda, i, e, g)$ is the intensity of light at wavelength, angle of incidence i, angle of reflection e and phase angle g (angle between direction of incident light and viewing direction); $m_S(i, e, g)$ is the geometric scale factor (determined by the illumination and viewing geometry) of $c_S(\lambda)$, the spectral power distribution of the specular component of the reflected light, and $m_b(i, e, g)$ and $c_b(\lambda)$ are the same quantities for the Lambertian reflection component. Specularities in the image are used to determine the weights for each component.

The Phong shading model [51] approximates the falloff in brightness of specular reflection as $\cos^n(\alpha)$, where α is the difference between the viewing angle and the angle of maximal specular reflectance. The value of n is determined empirically for a given surface, and varies from 1 (for matte surfaces) to 200 (for highly specular surfaces). At alpha = 0, the brightness is the maximum (i.e., 1), and falls off as the surface is rotated, to the minimum (i.e., 0) at -90^o and 90^o. The Phong model has been very widely applied by the computer graphics community as an effective method of achieving shading effects in rendering grey-scale and color images.

Nayar [46] describes the brightness of surface reflectance as a linear combination of the Lambertian and specular components (a concept similar to the Dichromatic Model [61]):

$$I = IL + IS,$$

where I is the total intensity at a given point in the surface, and IL and IS the intensities of the specular and Lambertian components. $IL = A \cos(\tau_S - \tau_n)$ (Lambert's law), where A is the constant representing the weight of the Lambertian component, and τ_S and τ_n are the directions of the illumination source and the surface normal. IS, modeled by the delta function [46], is $B\, \delta (\tau_S - 2\, \tau_n)$, where B is the weight of the specular component. Hence, $I = A \cos(\tau_S - \tau_n) + B\, \delta (\tau_S - 2\, \tau_n)$. The model is adapted for an extended light source to determine the weights of the reflection components by photometric sampling, a method by which brightness samples are obtained for multiple angles of illumination and viewing.

The above reflectance models show how the strength of the specular and Lambertian components of surface reflectance determines the effect of the illuminant color and the illumination and viewing geometry on the apparent color of a surface. Evidently, different types of surfaces exhibit different color shifts, depending on the combination of the aforementioned factors.

II.3.4 Shadows and inter-reflections

Inter-reflections and shadows can cause a further variation in color by altering the color of the light incident upon the surface [31]. Inter-reflections, for instance, cause light reflected off other surfaces in the scene to be incident upon the surface being examined. Shadowing can cause the elimination of incident sunlight (if the surface is self-shadowed), and further inter-reflection (if the surface is shadowed by another surface).

II.3.5 Imaging parameters

There are a number of imaging parameters, effective between the lens and the image plane, that may change the apparent color of objects in a scene. Clipping occurs when pixel values that are too high (i.e. too bright) or too low (i.e. too dark) are not registered beyond the limited response range of the camera, thus resulting in a loss of information and possible color skewing (if all three color bands are not clipped at the same time). Clipping is easily detected but not easily avoided, especially in outdoor images, where it is difficult for imaging hardware to adapt to the variation in the range of intensities. Any software approach to interpreting clipped pixels is bound to be ad-hoc and domain-dependent [47]; hence, until improvements in sensor design take place, machine vision methods may be forced to simply detect clipped pixels and discard those points in the image.

Blooming is a related phenomenon, where sensor cells saturated due to clipping "bleed" into neighboring cells. Blooming is harder to detect, except through finding clipped pixels and probabilistically tracing pixel values in the direction most likely to cause blooming [47]. On one hand, blooming is a much more serious problem than clipping because it is harder to detect; on the other, inter-cell bleeding is a simpler problem to prevent from a hardware design point of view. The method in this study does not present new approaches to classifying clipped or bloomed pixels; instead it assumes that clipping and blooming are localized phenomena and uses region-level heuristics (such as morphological operations and connected-components-based extraction of bounding boxes) to compensate for pixel-level errors introduced by the two phenomena.

Nonlinear response results in an inconsistent mapping between spectral power distributions and corresponding digital color values across the sensor range, and consequently a disproportional skewing in each of the color bands. For instance, the response to a surface highly saturated in the red channel (such as a red "Stop" sign) may be in the linear response range along the green and blue channels, but in the nonlinear range in the red channel. The effect of nonlinear response is virtually impossible to detect, except with careful calibration [47].

Another problem that has been shown to cause color skewing in calibration studies is chromatic aberration [5]. This phenomenon occurs because the focal length of a lens is a function of the wavelength of the light incident upon the lens. Hence, different colors may focus at different points with respect to the image plane and the optical axis. There are two types of displacement caused by chromatic aberration, lateral and longitudinal. Lateral chromatic aberration can cause light of a certain wavelength to focus on a cell neighboring the intended cell, causing color mixing. Experiments [5,47] indicate that this type of color mixing occurs mostly along surface boundaries, thus leaving the non-boundary pixels unaffected. Longitudinal displacement of light, i.e., along the optical axis can cause unequal blurring of different wavelengths. The same experiments [5,47] indicate that parametric methods sensitive to small perturbations in the assumed physics-based models are far more likely to be affected by such blurring than are the empirical methods used in this study, given the relative magnitude of color shifts due to the other complicating factors.

II.3.6 Overall distribution in RGB space

Assuming, from the standard image formation model [Horn], that apparent color is determined by the product of the incident light and the surface reflectance, and then somewhat altered by shadows, inter-reflections and imaging parameters, it can be deduced that the RGB distributions can be arbitrarily shaped, depending on the nature of the surface. The aforementioned reflectance models suggest that the distribution for a Lambertian surface will form a single thin region; that for a specular surface the distribution forms two clusters (one cluster near the color of the illuminant, due to the specular spike, and the other cluster near the color of lambertian component); and that the distribution for surfaces with mixed reflectance forms a continuous blob.

II.3.7 Proposed solution: Nonparametric classification

In principle, it should be possible to predict the apparent color of a surface in outdoor images, given the (i) sun-angle, (ii) weather conditions, (iii) surface orientation with respect to the sun and the camera, and (iv) robust models of surface reflectance. Since existing reflectance models have not been shown to be robust in unconstrained outdoor imagery, this approach assumes no knowledge of the aforementioned parameters; rather, the goal is to learn a function that maps RGB values from training samples of an object to particular classes. Thereafter, image pixels are classified into the separate classes based on the learned function associated with a given object. To classify pixels in outdoor color images, we need to select a non-parametric classification scheme that can approximate arbitrarily shaped functions in feature space.

There are two phases in the non-parametric approach to color recognition: training and classification. The training phase approximates a function (or a set of functions) representative of a distribution from samples of the distribution. The approximated function constitutes a mapping between a (training) set of RGB values and the surface (class) it represents. The classification phase determines the class of a given image pixel from the mapping function for the training set that pixel represents. Every pixel in a color image is classified, resulting in a gray-scale image where pixels belonging to a particular class will have the same gray value. There are two issues to consider: (1) the ability of the technique to generalize the function so as to adequately represent the distribution in color space without being too loose (resulting in the inclusion of samples from a different class), or too tight (resulting in the exclusion of samples belonging to the class); (2) the number of training samples required to approximate the distribution.

There are a number of techniques that have been used in other domains for function approximation and classification. In nearest-neighbor classification, given a set Xn = {X0, X1, ... , Xn} of n independent samples, a new instance Xn+1 is classified according to the distances between Xn+1 and each element of the set X0, X1, ... , Xn. The class assigned to Xn+1 is the class of the training sample with the shortest distance. The problem with using this approach on color images is that pixels forming a thin distribution will not be correctly classified using three-dimensional Cartesian distance in RGB.

Gaussian maximum-likelihood classification approximates an "average" feature value from samples and models the entire distribution as Gaussian noise about the average value; subsequently, pixels are classified based on the probability that they are noisy instances of the set represented by average. This technique cannot be applied to general outdoor color recognition because the variation of apparent color under daylight is not well-modeled as Gaussian noise.

Another way of classifying pixels is to segment the feature space and classify pixel instances based on their position in the segmented feature space. This can be done in a number of ways: by drawing explicit piecewise-linear boundaries in RGB space (decision trees [55,6a]); by learning a nonlinear function (genetic algorithms [45] and radial basis functions [53]) that maps RGB values explicitly to numerical values which are then thresholded to find decision boundaries; and by learning a mapping function that uses RGB as the input feature space but maps the input feature space to an intermediate feature space, so as the facilitate boundary fitting (neural networks with a hidden layer [57,16]).

(Univariate) Decision Trees [55] approximate a boundary by fitting hyperplanes around the samples, orthogonal to the axes of the feature space. Multivariate Decision Trees [6] are more general, and fit hyperplanes of arbitrary orientation around the distributions. Genetic algorithms (GA's) use principles from evolutionary biology to converge on optimal parameters of a fixed-dimensional nonlinear polynomial function. Radial basis functions (RBF's) approximate a function as a weighted sum of Gaussians. Neural Networks (NN's) are more general than GA's or RBF's, and approximate a function of arbitrary dimensionality (determined by the number of hidden units) as a weighted sum of nonlinear squashing functions. Although NN's can be expected to perform well for RGB distributions in this application, the arbitrary nature of the hidden layer feature space makes analysis difficult; consequently, the work presented here uses Multivariate Decision Trees.

II.4 Previous Work on Multivariate Decision Trees

In a seminal paper from the machine learning community, [6] opposed combining traditional linear classifiers known as linear machines (described in [25] and elsewhere) with the recursive segmentation process found in decision trees [55]. Basically, linear machines divide feature space via hyperplanes to correctly classify as many instances as possible. Decision trees traditionally divided feature space according to a single thresholded feature and then recursed, redividing each section until feature space was segmented into small areas, each of which contained instances of only a single object class. By combining these two methods, Brodley & Utgoff [6] created a powerful system for creating a piecewise-linear division of feature space for classification purposes.

As part of the Rome Labs research, we became the first researchers to test multivariate decision trees (as the combined approach is now known) in a computer vision domain [8]. What we discovered surprised us -- for the task of labeling pixels in outdoor color imagery, the MDTs outperformed not only traditional Gaussian techniques such as minimum distance classifiers (which we expected), but also other non-parametric approaches such as neural nets. Trying to understand this phenomenon led us to investigate the properties of color in outdoor scenes, and led to the thesis work of Shashi Buluswar [10]. Although this work is on-going, our current understanding is that the shifts in apparent color of objects in outdoor scenes are caused primarily by shifts in the color of daylight (due to geospatial position, time of day and weather), and by the ratio of direct sunlight to ambient skylight. These factors cause color shifts that are linear in Hue/Saturation space, and form a smooth parabola in RGB color space. Because MDTs are constrained to piecewise-linear approximations in feature space, they tend to fill in a close approximation to this parabola even when only very little training data is available, whereas neural nets (which classify in a higher-dimensionality space) need far more training data to distinguish the parabola from other, incorrectly shaped regions in feature space, that the MDT never has to consider. In other words, the decision surface used by

MDTs is more appropriate to this task than the higher-dimensionality surface used by neural nets.

III. Real-time color classification in RSTA UGV domain

III.1 Learning to compensate for natural chromatic variation in outdoor images

As discussed above, a technique has been developed that uses training images of an object under daylight to learn the shift in apparent color of the object. This method uses multivariate decision trees for piecewise linear approximation of the region corresponding to the object's appearance in color space. Pixels in outdoor scenes can then be classified depending on whether they fall within the appropriate region in feature space. Clusters of target pixels are then clustered into regions of interest (ROIs) for a model-based RSTA system. This general approach has been applied to detection of camouflaged vehicles in outdoor scenes. Based on the implications of the results of this work, we have demonstrated a technique that is capable of accurately modeling the color shift of an object under daylight in order to predict its appearance under particular weather conditions at given times of the day.

III. 2 Real-Time Detection of Camouflaged Vehicles

We tested the ability of multivariate decision trees (MDTs) in the important application of automatic target detection (ATD) to identify military vehicles in the Ft. Carson data set [8]. This is obviously a difficult task: not only are these images of outdoor scenes with natural lighting, but the vehicles have been intentionally camouflaged! In addition, the classifier is using only the local RGB values of the pixels.

The motivation for attempting such a challenging task is the need for a rapid focus of attention (FOA)mechanism for RSTA. Any classification scheme that labels pixels based on only their RGB values can be loaded into a 24-to-1 bit lookup table, and applied in real time. Obviously, this feature set is so restricted that some errors are inevitable, but if many pixels can be accurately classified, then contiguous groupings of target pixels can be used as regions of interest (ROIs) for the RSTA identification and matching algorithms to process. The vital question is whether pixels can be classified accurately enough to produce "useful" ROIs.

Tests were conducted on a set of 44 images from the Fort Carson data set. (This is essentially all the RGB images from the first CD except for calibration images and a few images that are excessively dark; [8] describes the complete data set and how it can be obtained.) The objects being sought were camouflaged military vehicles in a natural setting. There are four different army vehicles in these images: two armored personal carriers, a pickup truck, and an M60 tank; all four bear slightly different colors and patterns of camouflage.

Quantitative evaluation of the performance of the classifier was done at two levels: pixel and ROI. At the pixel level, classification accuracy is a poor evaluation criterion; since 99.6% of the pixels are background, a classifier that labeled nothing as target would be highly accurate. Instead, we evaluate our system separately in terms of its accuracy on target pixels and its accuracy on background pixels.

The pixel-wise performance of the MDT classifier on the Ft. Carson images is given in Table 3. On the average, 53.40% of the target pixels and 97.50% of the background

pixels were correctly classified. These results were obtained using cross-validation, where the decision tree was trained on half the images and tested on the other half. In order to train the decision trees, approximately 35,000 sample pixel values were extracted from the training images. Figure 10 shows two examples of source, classification, and result images.

Table 3. Ability of Multivariate Decision Trees to separate target pixels by image (of camouflaged vehicles) from background pixels for the Ft. Carson images [8].

	% of Target Pixels Correctly Labeled	% of Background Pixels Correctly Labeled
Average	53.40	97.50
Best	84.66	99.67
Worst	31.87	94.71
SD	10.39	1.62

Table 3 provides a measure of the multivariate decision tree as a classifier for this domain, testing the argument about MDT' s ability to compensate for natural chromatic variation due to varying lighting conditions, etc. Another way to test the validity of this argument is to look at the distributions of target and background pixels in the Ft. Carson data, and the region of color-space associated with target pixels learned by MDT. Figures 11 and 12 show the target and background pixels in a chromaticity diagram; Figure 13 shows the region of space learned by the MDT, also projected onto a chromaticity diagram. Note that large portions of the space are occupied by neither target not background pixels, so the label associated with these parts of color space is arbitrary. In this example, MDT assigns much of this space the class "target".

A region-level evaluation, on the other hand, certifies how well MDTs perform at generating regions of interest for a RSTA system. ROIs were generated from the pixel-label data by finding connected groups of target pixels that were at least ten pixels high and wide, and that had a maximum aspect ratio (height-to-width or width-to-height) of 10:1. ROIs whose bounding rectangles overlapped were then merged. For ROIs, there are two relevant evaluation criteria: the probability that a ROI will be generated for all targets, and the probability given an ROI that it contains a target. For the Ft. Carson data, there were 112 occurrences of targets in the 44 test images. The MDT classifier produced 153 ROIs, of which 109 overlapped the targets; only 3 targets were not recognized, and there were 44 false alarms. Table 4 shows the performance of MDT at generating ROIs.

Table 4. Ability of MDT to generate regions of interest around target vehicles in the Ft. Carson data.

	Targets	Targets Found	False alarms
Total	112	109	44
Best	4	4	0
Worst	4	3	3

The experiments shown here are very promising. MDTs appear to work well on a difficult set of outdoor scenes. The accuracy in terms of generating regions of interest is

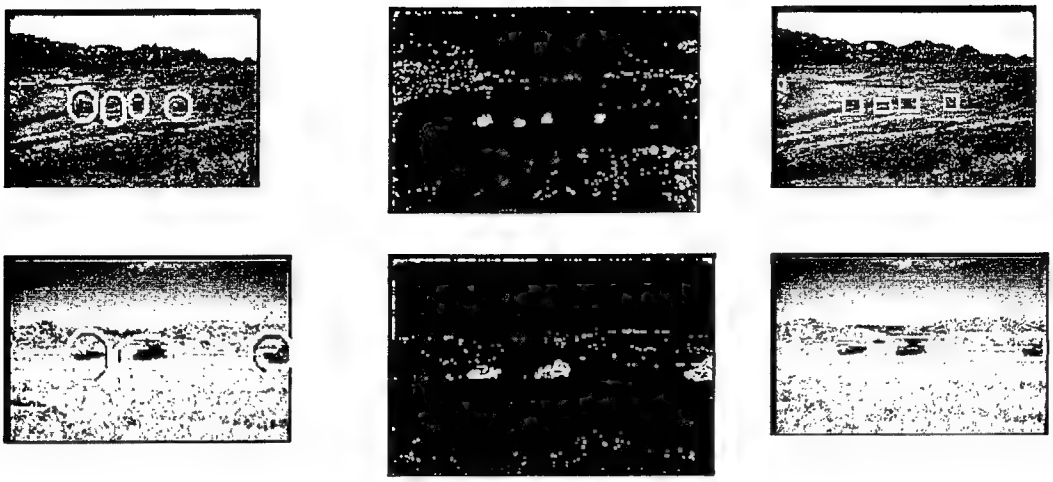


Figure 10. Results from MDT-based classification for camouflaged target detection - original color images (left, targets marked with circles), binary classification (middle), targets extracted (right).

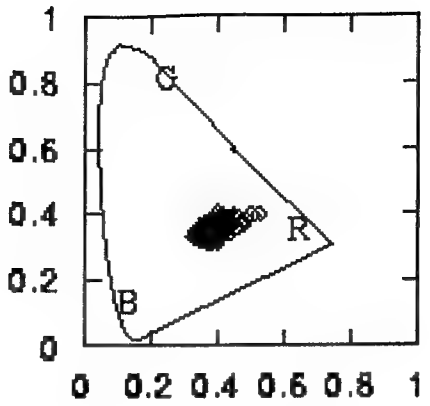


Figure 11. Target pixel distribution

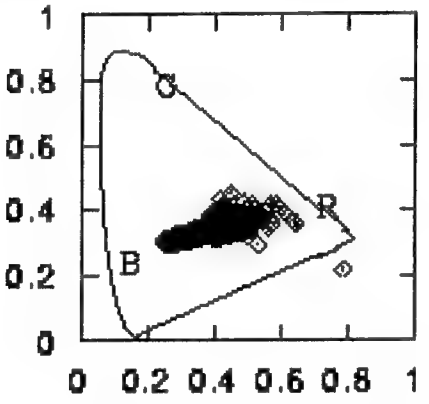


Figure 12. Background pixel distribution.

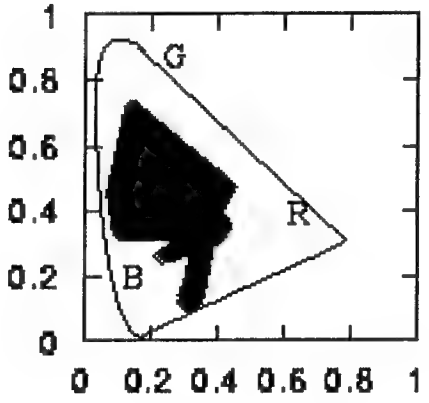


Figure 13. The region of color space (projected onto a chromaticity diagram) labeled "target" by the MDT.

high enough to suggest that MDTs are an immediately useful technology for color-based object recognition.

III.3 Outdoor Color Recognition for IVHS

MDT-based classification is currently being used in the National Automated Highway System (AHS) project for detecting lanes and obstacles in highway scenes for autonomous vehicles. Figure 14 shows a sample image from a highway scene; the goal in this application is to find the lane-markers and obstacles. There are two lookup tables constructed, one for lane-markers and one for the road surface. Pixels classified as non-road are either lane-markers or obstacles; lane-marker pixels are classified separately, thus identifying the objects on the road that are potential obstacles. The vehicle heading is determined by fitting lines to the lane-marker pixels, and the potential obstacles are extracted by clustering connected pixels and region-level heuristics. In this system, stereo and motion techniques are used to further prune the obstacle map by identifying the potential obstacles that lie above the ground plane. Representative results of classification for lane-markers and obstacles are shown in Figure 15. The color-based component of the obstacle detection system has been tested on thousands of images of hundreds of sequences, and tests conducted in the AHS project have found the system to be sufficiently "reliable" for practical application.

III.4 Off-road Classification for UGV

While the goal of the AHS project is to provide highway-based autonomous vehicles, the U.S. military is interested in autonomous off-road driving systems. Toward this end, the Unmanned Ground Vehicle (UGV) project developed vehicles that used stereo cameras to detect obstacles by marking all objects over a fixed height above the ground plane (corresponding to the ground clearance of the vehicle) as obstacles. In the off-road tests in Colorado, this strategy proved excessively conservative, in that it forced the vehicle to meticulously avoid yucca bushes and other "obstacles" it could easily drive over. In this scenario, MDT-based classification was used to detect yucca bushes and eliminate them from the obstacle map. In 45 test images that contained 212 identifiable yucca bushes, 176 of the bushes were successfully detected; there were many false positives at the pixel level, mostly from grassy regions, which did not affect the performance of the system because they were not in the initial obstacle map. Figure 16 shows results from one image with a simulated obstacle map; the yucca bushes (pixels) are detected, and those pixels are removed from the obstacle map, leaving only the rocks in the final obstacle map.

IV. Motivation for real-time interactive classification

An important early component of many vision systems is initial pixel classification. With pixels classified by likely region type, higher levels of reasoning can make useful inferences about the contents of the image. However, the construction of pixel classifiers is a labor-intensive task involving user interaction to manually select feature sets, manually select local training data for each desired object class, and then to provide feedback as a result of classification for additional refinement until satisfactory global classification is achieved. For example, in the aerial domains we have explored this involved classification of foliage, roads, grass-covered ground, etc. The typical specification of training data is a painstaking, arduous task, involving the close-up examination of images, and while crucial it is not a very pleasant process. An additional concern is the determination of when the quantity and accuracy of the training data is sufficient for effective classification.

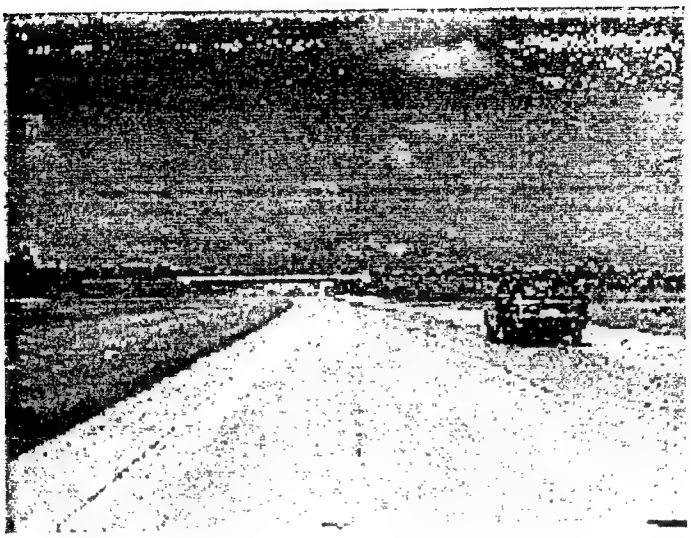
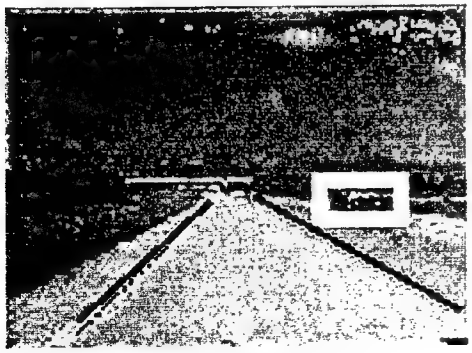


Figure 14. Sample image from a typical highway scene. The goal is to find lane markers and obstacles.

(a)

(b)



(c)

Figure 15. Representative results for MDT-based classification for lane-markers and obstacles (left to right, top to bottom) - (a) classification for lane-markers, (b) classification for road, and finally (c) detected obstacles and lanes.

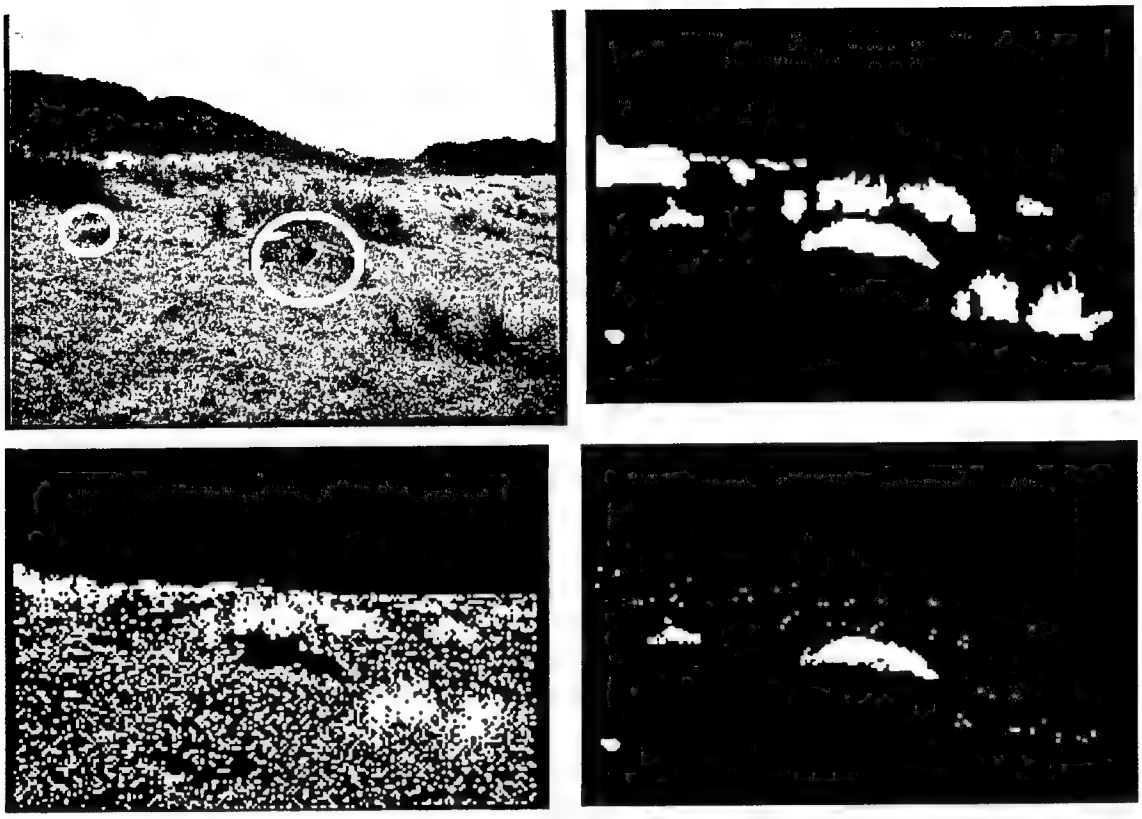


Figure 16. Results from MDT-based classification for yucca bushes (left to right, top- to-bottom) - original color image (rocks/obstacles marked with circles), simulated depth-based obstacle map, classification for yucca, final obstacle map.

We are in the process of developing an exciting new approach to this fundamental but often overlooked problem within the domain of aerial imagery. The labor associated with manual specification of data can be reduced by providing fewer, but highly informative training instances, and rapid user feedback to direct the further specification of training data. If one could be informed during the construction process where the classifier would make mistakes, one could generate an informative training instance by providing a correct label for a currently misclassified pixel, and additional training data for statistically impoverished training states. Thus, the prototype system we have implemented is an exploration into a new classification paradigm. This suggests an interactive real-time tool, which immediately narrows the choice of a classification algorithm to one that is efficient and incremental, since this is usually a computationally expensive and off-line process.

We have developed an exciting prototype interactive tool [52] that allows the user to immediately see the result of selecting incremental training data so that he can adjust the further selection on the basis of inaccurate classification. This system shows that the incremental classifier converges to satisfactory performance after a very small number of training instances and required only a fraction of the typical human effort to provide them.

The current 2D version of our interactive classifier is being developed into an even more exciting version, a modern high performance 3D visualization tool for incremental classification of terrain in aerial images. By applying the UMass terrain reconstruction system (Terrest)) [59,60] to a pair of aerial images, a three-dimensional world model can be created and applied in a fly-through visualization system. This now allows interactive training of the classifier with the user examining the world data from more natural and understandable viewpoints that show the sensor data in the context of its three-dimensional characteristics, e.g. from a 45 degree downward oblique view, where sides of objects and terrain are more understandable. During this fly-trough the operator will be able to select training data for the classifier interactively. The ability to see a three-dimensional model (possibly utilizing 3D stereo goggles) will make it easier for a user to utilize additional background knowledge and context. Classification results can then be rapidly overlaid onto the terrain model using a variety of graphic display techniques, and with incremental real-time updating of training data. The user will be able to continue to roam across the terrain, immediately seeing the result of the classifier to this point. Multi-resolution processing strategies that work with higher accuracy close to the location of the observer (i.e. near the current screen for the visualization) can be implemented to decrease latency. When a global view is desired the operator can move far above the surface to see the whole area. Support for this real-time visualization using a high-performance multiprocessor is currently underway, and there are discussions underway to transfer this capability to the Army Research Labs.

Our supervised learning model for pixel classification is shown in Figure 17. The teacher is a human who operates a graphical user interface. He/she can select images for training and, for any image, select and label small clusters of pixels. The learner is a computer program that communicates with the teacher's interface. As we have pointed out, the fundamental aspect of this model is that it is interactive. The teacher does not need to provide a large number of instances a-priori that may or may not be informative. Instead, each time the human provides a new instance, the learner revises its classifier as necessary, and then communicates the class labels for all pixels of the image. This lets the human see the misclassified pixels immediately.

At the heart of this system is an incremental decision tree inducer (ITI) algorithm [65,66]. ITI revises its tree incrementally, meaning that it can accept training instances serially,

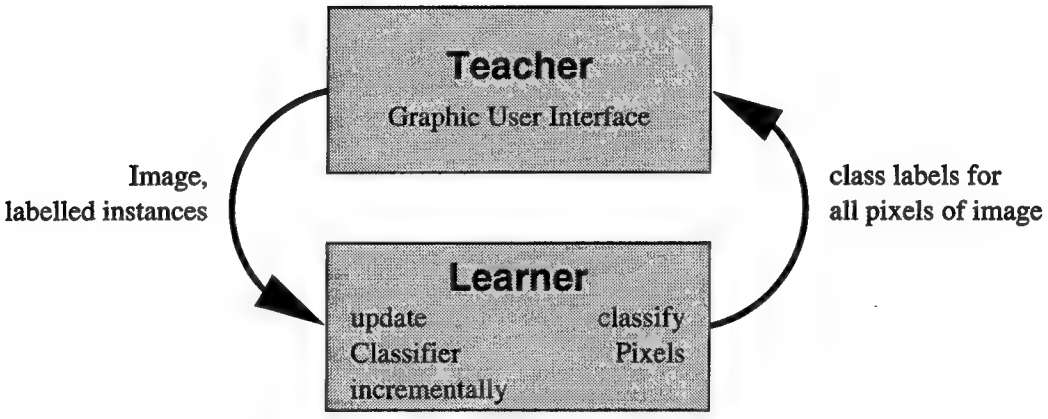


Figure 17: Structure of the learning classifier system. The arrows indicate principal data flow channels.

without needing to completely rebuild the tree repeatedly. This means that it can reorganize an existing classification tree rapidly in a (hopefully close to) real-time manner.

IV.1 A Sample Session

Here we present an example session Figure 18. The goal is to learn to classify pixels in an outdoor house image as one of four classes: sky, roof, brick, or foliage. Pixels that belong to another region type, e.g. the pavement class, are assumed not to be of interest. They will be classified as any of these four classes, but we will not care in this experiment. Otherwise, a class of "other" can be defined with additional training data from all classes that are not the 4 desired classes. For simplicity, six features have been used, which are the RGB values of a pixel, and the variances of each in a 3x3 window centered around that pixel. Each mouse click by the teacher creates a 3 by 3 square of training instances (labeled pixels) at that location and can be dragged through the image to select larger amounts quickly.

Figure 18(b) shows the classification result after labeling just one square of each of two classes. The sky is already almost perfectly separated from the rest of the image. In Figures 18(c,d) one square of each of the remaining two classes is added. While the addition of the brick class again results in good generalization, things become more complicated when a sample of the foliage class is added. This occurs because in this image the foliage and roof classes are mainly characterized by large variances within the RGB intensities, rather than the color themselves (i.e. hue and saturation), and thus hard to separate. The image in Figure 18(i) contains several contradictory training instances that belong to different classes, but are indistinguishable using the given feature set. Therefore, perfect classification is not achievable, given these features and training samples.

IV.2 Assessment of Effectiveness of the Methodology

The error-driven point-click loop is very productive. It is much more satisfying to construct a classification system by correcting its mistakes than it is to generate large amounts of meticulously constructed training data in an attempt to guarantee that all are training samples are appropriately informative. In terms of the accuracy and efficient production of well chosen training instances, our experience is that the system performs very nicely. Because our experience has shown that the tree generalizes well using this methodology, it is not necessary to make a large number of training and classifier corrections.

In terms of computational efficiency, ITI produced small trees that were highly efficient classifiers. The learner classifies a small image of about 200 X 200= 40K pixels in a few seconds. This rate is linear in the number of pixels in an image and largely independent of the number of features, but obviously depends on the time required to compute the features.

In this application, we, the teacher, were not concerned with some of the issues that are important within machine learning research. The teacher just wants a reasonably small tree (to produce a fast classifier) obtained with an acceptably small amount of training. If a resultant decision tree is somewhat larger than optimal, and therefore required a few more training examples than might otherwise be necessary, yet was produced much more quickly, then the cost of the overall task of producing the pixel classifier might be far less.

Figure 18: An example session on a small image (a), 112 by 115 pixels. In the following images, the class labels are represented by mnemonic (but otherwise meaningless) colors: SKY blue, ROOF red, BRICK yellow, and FOLIAGE green. The tiny bright squares are the training instances provided by the teacher, and the darker areas of corresponding colors are the classification results of the learner. Images (b)–(d) show the results after adding one set of training instances for each class, and (e)–(i) are snapshots during some refining.

While learning is very fast in the early stages of training, the later stages usually involve a lot of refinement. As errors are corrected in one part of the image, others appear in different areas. This usually goes on for some number of iterations. On the other hand, inspection of the decision tree with respect to contradictory training instances can reveal when and where the discriminatory power of the feature set on the selected training examples are reached. By choosing appropriate training examples, or changing the priors of the classes, one can bias the classifier to some extent to avoid certain types of errors while tolerating others. For example, in the illustrations above it was not possible to completely separate the roof class from the foliage class. If the teacher were more concerned about correct classification of the roof than the tree, he could give the system more training examples of the roof class, which would increase the accuracy of roof pixels at the expense of more misclassifications of tree class pixels.

IV.3 Quantitative Results

In this section, we compare our Learning Classifier with a previously published classification result [70]. We chose this example because it uses state-of-the-art techniques, the task is realistic, and their data and results were readily available to us.

Wang et al. [70] considered an orthographic aerial image show this image? of a rural area in Ft. Hood, Texas. The goal was to build a pixel classifier to recognize the four terrain classes bare ground (road, riverbed), foliage (trees, shrubs), grass, and shadow. Their most effective feature set consisted of 12 co-occurrence features (angular second moment, contrast, and entropy at four angular orientations each; [74], four three-dimensional features introduced by Wang et al. [70], and the intensity. The co-occurrence features employed have previously been claimed to be highly effective for classification [77,72, 71,76], and the 3D features were shown to be highly discriminative in this task. The Foley-Sammon transform [73] was employed as a classifier. FST is a linear discriminant method which is considered effective [75,77].

As a training set, Wang et al. [70] used four homogeneous chips of different sizes: 99x99 (foliage), 75x75 (grass), 37x37 (bare ground), and 11x11 (shadow). These 16916 training pixels constitute less than 1 of the entire image (1936789 pixels). Ground truth was generated by hand. The achieved classification accuracy was 83.4 percent [70].

To provide a baseline of the performance of ITI with respect to FST on this task, we ran ITI (in the conventional batch mode) on the same input data as described above, and achieved a classification accuracy of 85.9 percent. An image of the result is shown in Figure 19.

We then trained a classifier interactively on this data, as explained above. Each mouse click produced a 3x3 blob of training instances of the same class, as in the previous example. To show the behavior of the learning process, a series of 100 mouse clicks was performed, yielding a total of 900 training instances at the end of the session. After each mouse click, the decision tree was used to reclassify the displayed image, as explained above. Such a training procedure takes typically less than 15 minutes.

Each of the 100 decision trees thus generated was saved to a file. At the end of the session, they were used off-line to evaluate the accuracy of the decision tree after each mouse click by comparing the classification results with Wang et al.'s [70] ground truth data. These results are plotted in Figure 20.

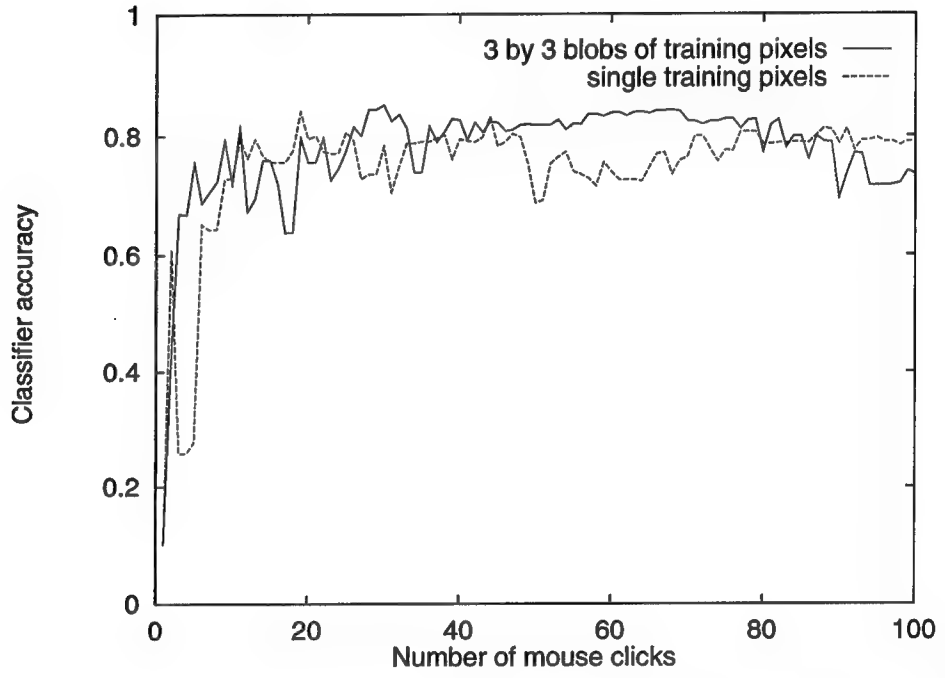


Figure 19: Plot of classification accuracy versus number of mouse clicks during interactive training of a classifier.

We also ran a training session where each mouse click generated just one single training instance. The performance curve is also shown in Figure 20. An image of the results is shown in Figure 21. All quantitative results are summarized in Table 5. It is striking how rapidly the classification accuracy of the baseline classifier could be closely approximated. The increase in classifier quality is dramatically demonstrated by the small size of the decision trees, and the small numbers of features actually used.

Table 5: Summary of classification

	interactive			traditional
mouse clicks:	19	30	68	(not applicable)
Training instances:	19	270	612	16916
correct:	84	85	84	86
tree nodes:	9	25	65	181
max. tree depth:	3	10	18	23
features used (of 17):	3	7	11	16

The performance curves in Figure 20 show that good classification accuracy is achieved after very few training examples. However, the accuracy does not grow monotonically: Often, after correcting one misclassification by giving a counterexample, the accuracy drops dramatically. Usually, very little additional training is sufficient to recover from such a drop. Note also the long stable region (peak at 68 mouse clicks) of the blobs training curve.

The images show that ITI's misclassifications were benign. Most of them occurred in areas where it is hard even for a human to assign a fixed class label to a single pixel. Here, an important limiting factor of assessing classifier performances lies in the 'crispness' of the ground truth data.

We have demonstrated a new methodology for interactive training of pixel classifiers. It is a very effective tool for selecting few but informative training instances, resulting in great reduction of human labor and dramatically simplified classifiers. A simple user interface allows training with useful real-time feedback on images of virtually unlimited size.

To build interactive learning systems that update their parameters in real time, incremental learning algorithms are beneficial. The ITI classifier works well in our application. Fast incremental learning algorithms are an open area of research with many potential applications for interactive learning systems.

V. New Formulation for Learning Control Policies based on Markov Decision Processes and Reinforcement Learning

The original paradigm for SLS was to use decision trees (or other classifiers) to evaluate intermediate data results at each level of representation, and to use explanation-based learning to choose how to transform data from one level of representation to the next. Although successful in some trials, this approach suffered from two problems. One problem was that the features had to be discretized, as discussed in Section 1. This problem might or might not be solvable within the original paradigm. However, the second problem was more severe: the explanation-based learning algorithm was looking for logical consistency as a basis on which to choose algorithms. Unfortunately, computer vision algorithms are often inconsistent, and we soon discovered that when applied to large training sets, the system would often find no logically consistent solution.

Transformational Proc.	Input	Output	Ref
Line Extraction	Image	2D Lines	[7]
Region Segmentation	Color Image	Segmentation	[5]
Multivariate Decision Tree	Segmentation	Regions	[8, 14]
Region Merging	Regions	Region	
Interest Operator (Anandan)	Image, Region	2D Points	[2]
Interest Operator (Moravec)	Image, Region	2D Points	[24]
Min. Dist. Classification	Region	Label	
Polygonal Approximation	Region	2D Lines	
Line Extension	2D Lines	2D Line Groups	
Rectalinear Line Grouping	2D Lines	2D Line Groups	[28]
Pencil Extraction	2D Lines	Line Pencils	[13]
Vanishing Point Analysis	Line Pencils	3D Orientation	[13]
Convex Hull	Line Pencil	Region	
Trihedral Jnct Finder	2D Lines	Trihedral Jnct	
Trihedral Angle Analysis	Trihedral Jnct	3D Orientation	[22]
Planar Distance (Scale)	Region or 2D Points, 3D Orientation	3D Pose	
Subgraph Isomorphism	2D Line Group	Correspondence	[29]
Image Resection	2D Points, Correspondence	3D Pose	
Perspective Projection	3D Pose	2D Points	
Geometric Matching	2D Lines	3D Pose	[6]

Table 6: *Transformational Procedures (TPs). Note that the Input column lists only image-based arguments. Model-based arguments are not listed for Subgraph Isomorphism, Geometric Matching and Planar Distance TPs.*

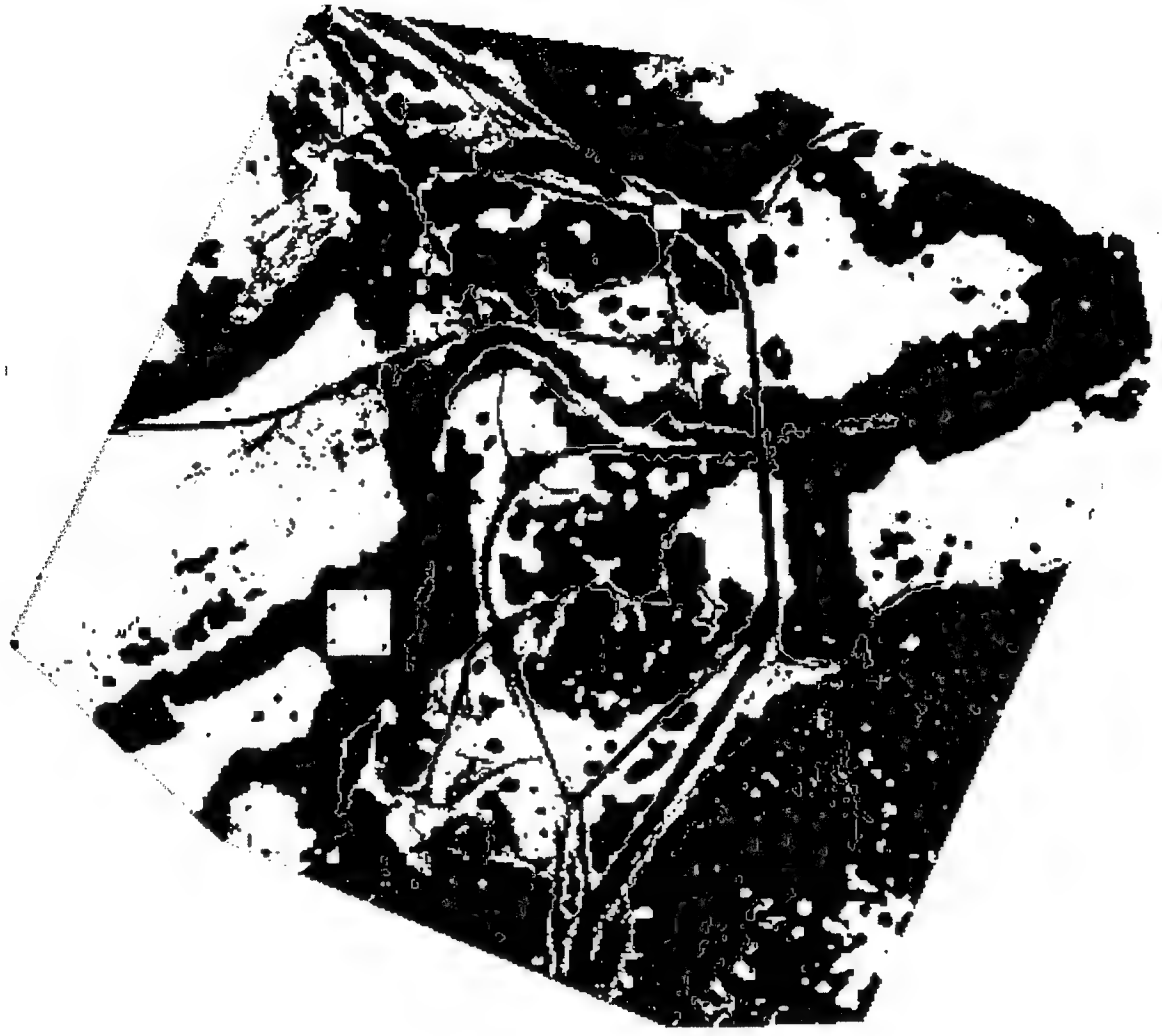


Figure 20: Classification results using ITI in batch mode on the same training set as in Wang et al., overlaid on a ground truth image (1803 × 1591 pixels). The class label of a pixel is visualized by generating a false color, where the hue corresponds to the class, and the intensity corresponds to the gray scale intensity of the underlying ground truth image. No classification results are provided for the training pixels, which constitute the uncolored square areas. In most cases, hue boundaries coincide with intensity boundaries, indicating accurate classification. Misclassified areas are surrounded by boundaries where only one of hue or intensity changes.

Figure 21: Classification results using the Learning Classifier.

This is why we never reported results on the original version of SLS over large training sets.

What we needed was a learning algorithm that would select actions based on their expected value, rather than logical consistency. We found such an algorithm in reinforcement learning, which trains control policies (i.e. control strategies) so as to maximize the expected value of the action selected at each step.

Unfortunately, the reinforcement learning model as generally applied requires a discrete state space, and we have already noted that discretizing the features results in a loss of information. We therefore developed a model for applying reinforcement learning to object recognition based loosely on work by [63] and [69] in which each level of representation is viewed as a continuous feature space defined by its measurable attributes, and control policies are learned using a combination of reinforcement learning and neural networks that map points in the feature spaces onto optimal actions.

An initial implementation of this new approach was completed in December, 1995. In the first major test of this system, it was 10-for-10 in recognizing rooftops in aerial images of Ft. Hood, TX. These results (which use a reduced library of visual procedures) were reported at the ARPA Image Understanding Workshop at Palm Springs, CA in February [21]; additional experiments were reported at the International Conference on Pattern Recognition in Vienna in August [20].

In addition to more robust performance, there are theoretical advantages to using the reinforcement learning (RL) framework. Because RL is based on a Markov Decision Process model, vision systems based on RL should be more easily integrated with robotics and other traditional control systems. Furthermore, since the neural networks within the RL framework learn what are known as Q-functions in control theory, the RL system is able to provide the user with feedback about the expected accuracy and remaining cost of the interpretation process at any point. Such information is useful not only to human users, but also to any automatic planning systems that might by used to schedule or allocate sensory resources. A theoretical understanding of the implications of applying reinforcement learning to computer vision is as much a goal of the on-going research as is the prototype system.

The reinforcement learning version of this work is potentially a major breakthrough in the area of visual control, and we are pleased to say the DARPA has decided to fund Draper in the continuation of this work at Colorado State University as part of the Automatic Population of Geospatial Databases (APGD) program.

VI. References

[1] J.R. Beveridge, A. Hanson, and D. Panda, "RSTA Research of the Colorado State, University of Massachusetts and Alliant Techsystems Team, *Proc. ARPA Image Understanding Workshop,* Monterey, CA, November 1994.

[2] J. Ross, Beveridge, Griffith, Joey, Kohler, Ralf, R., Hanson, Allen R., and Riseman, Edward M., "Segmenting Images Using Localized Histograms and Region Merging", *International Journal on Computer Vision,* 2, pp. 311-347, 1989.

[3] M. Bhandaru, B. Draper and V. Lesser. "Learning Image to Symbol Conversion," *AAAI Workshop on Machine Learning in Computer Vision,* Raleigh, N.C., Oct. 1993. pp. 6-9.

[4] M., Boldt, Weiss, R., and Riseman, E., "Token-Based Extraction of Straight Lines", *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 19, No. 6, November/December 1989, pp. 1581-1594.

[5] T.E. Boult and G. Wolberg, "Correcting Chromatic Aberrations Using Image Warping", *DARPA Image Understanding Workshop,* 1992.

[6] C. Brodley and P. Utgoff, "Multivariate Decision Trees", *Machine Learning,* 19:45-77, 1995.

[7] J., Brolio, Draper, B., Beveridge, J. Ross, and Hanson, R., "The ISR: A Database for Symbolic Processing in Computer Vision", *special issue of Computer, titled Image Database Management,* December 1989, pp. 22-30

[8] S. Buluswar and B. Draper, "Non-parametric Classification of pixels of Varying Outdoor Illumination," *ARPA Image Understanding Workshop,* Monterey, CA., Nov. 1994, pp. 1619-1626.

[9] S. Buluswar and B. Draper. "Color Recognition by Learning: ATR in Color Images," *IEEE Conf. on Computer Vision and Pattern Recognition,* Puerto Rico, June 1997.

[10] S. Buluswar, "An Analysis of Outdoor Color Vision", forthcoming Ph.D. dissertation, University of Massachusetts.

[11] G. Buchsbaum, "A Spatial Processor Model for Object Color Perception", *Journal of the Franklin Institute,* 310:1-26, 1980.

[12] P.R. Cohen, and Feigenbaum, E.A. (eds)., The Handbook of Artificial Intelligence, Vol. 3, Los Altos, CA, William Kaufman, Inc., 1982.

[13] Collins, R., Hanson, A., Riseman, E., Jaynes, C., Stolle, F., Wang, X., and Cheng, Y.Q., "UMass Progress in 3D Building Model Acquisition", Proc. *ARPA Image Understanding Workshop,* Vol., I, pp. 305-315, Palm Springs, CA, February 1996.

[14] R., Collins, and Weiss, R., "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere', *IEEE Third International Conference on Computer Vision,* Osaka, Japan, December 1990, pp. 400-403.

[15] J. Crisman and C. Thorpe, "Color Vision for Road Following", Vision and Navigation: The Carnegie Mellon NAVLAB, Kluwer, 1990.

[16] J.E. Dayhoff, Neural Network Architectures, Van Nostrand Reinhold, New York, 1990.

[17] B. Draper, A. Hanson and E. Riseman, "Knowledge-Directed Vision: Control, Learning and Integration", *Proceedings of the IEEE,* 84(11):1625-1637, (Nov.) 1996.

[18] B. Draper, C. Brodley and P. Utgoff, "Goal-directed Classification Using Linear Machine Decision Trees", *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 16(9):888-893 (1994).

[19] B. Draper, "Learning Control Strategies for Object Recognition," in Symbolic Visual Learning, K. Ikeuchi and M. Veloso (eds.), Oxford University Press, New York, to appear.

[20] B. Draper. "Object Recognition as a Markov Decision Process," *International Conference on Pattern Recognition,* Vienna, Austria, Aug. 25-29, 1996, Vol. IV, pp. 95-99.

[21] B. Draper, "Learning Grouping Strategies for 2D and 3D Object Recognition", *ARPA Image Understanding Workshop*, Palm Springs, CA, 1996, pp. 1447-1454

[22] B. Draper, "Learning from the Schema Learning System," *AAAI Workshop on Machine Learning in Computer Vision,* Raleigh, N.C., Oct. 1993, pp. 75-79.

[23] B. Draper, "Learning Object Recognition Strategies", Ph.D. dissertation, Computer Science Dept., Univ. of Massachusetts, TR93-50, May 1993.

[24] B.A., Draper, Collins, R.T., Brolio, J., Hanson, A.R., and Riseman, Edward, M., "The Schema System", *International Journal on Computer Vision,* 2, pp. 209-250, 1989.

[25] R.O. Duda and P.E. Hart, Pattern classification and scene analysis, New York: Wiley, 1973.

[26] M. D'Zmura, and G. Iverson, "Color Constancy: Basic Theory of Two Stage Linear Recovery of Spectral Descriptions for Lights and Surfaces", *Journal of the Optical Society of America,* A 10:2148-2165, 1993.

[27] G.D. Finlayson, B.V. Funt and K. Barnard, "Color Constancy Under Varying Illumination", *Proceedings of the Fifth International Conference on Computer Vision,* 1995.

[28] D. Forsyth, "A Novel Approach for Color Constancy", *International Journal of Computer Vision,* 5:5-36, 1990.

[29] W. Freeman and D. Brainard, "Bayesian Decision Theory: the maximum local mass estimate", *Proceedings of the Fifth International Conference on Computer Vision,* 1995.

[30] B.V. Funt and M.S. Drew, "Color Space Analysis of Mutual Illumination", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12:1319-1326. 1993.

[31] R. Gershon, A. Jepson, and J. Tsotsos, "The Effects of Ambient Illumination of the Structure of Shadows in Chromatic Images", RBCV-TR-86-9, Dept. of Computer Science, University of Toronto, 1986.

[32] A. Hanson and E. Riseman, Computer Vision Systems, Academic Press, December 1978.

[33] F.S. Hillier, and Lieberman, G.J., Introduction to Operations Research, San Francisco: Holden-Day, Inc., 1980.

[34] B.K.P. Horn, Robot Vision, MIT Press, Cambridge, MA, 1987.

[35] D. Judd, D. MacAdam, G. Wyszecki, "Spectral Distribution of Typical Daylight as a Function of Correlated Color Temperature," *Journal of the Optical Society of America,* 54(8):1031-1040, 1964.

[36] K., Kanatani, "Constraints on Length and Angle," *Computer Vision, Graphics, and Image Processing,* 41:28-42, 1988.

[37] G.J. Klinker, S.A. Shafer and T. Kanade, "Color Image Analysis with an Intrinsic Reflection Model", *Proceedings of the International Conference on Computer Vision,* 1988.

[38] R., Kumar, and Hanson, A., "Robust Estimation of Camera Location and Orientation From Noisy Data Having Outliers", *IEEE Workshop on Interpretation of 3D Scenes,* Austin, TX, November 1989, pp. 52-60.

[39] G. Kutlu, B. Draper and E. Moss, "Support Tools for Visual Information Management," *Fifth Annual Symposium on Document Analysis and Information Retrieval,* 1996.

[40] G. Kutlu, B. Draper, E. Moss and E. Riseman, "Persistent Data Management for Visual Applications," *ARPA Image Understanding Workshop,* Palm Springs, CA, 1996, pp. 1519-1523.

[41] G. Kutlu. "Database Techniques in Computer Vision Software Development", forthcoming Ph.D. dissertation, University of Massachusetts.

[42] E.H. Land, "Lightness and Retinex Theory", *Scientific American,* 237(6):108-129, December 1977.

[43] S.W. Lee, "Understanding of Surface Reflections in Computer Vision by Color and Multiple Views", Ph.D. Dissertation, University of Pennsylvania, 1992.

[44] L.T. Maloney and B.A. Wandell, "Color Constancy: A Method for Recovering Surface Spectral Reflectance", *Journal of the Optical Society of America,* A3:29-33, 1986.

[45] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

[46] S.K. Nayar, K. Ikeuchi, and T. Kanade, "Determining Shape and Reflectance of Hybrid Surfaces by Photometric Sampling", *IEEE Transactions on Robotics and Automation,* 6:418-431, 1990.

[47] C. Novak, S. Shafer and R. Wilson, "Obtaining Accurate Color Images for Machine Vision Research", *Proceedings of the SPIE,* vol. 1250, 1990.

[48] C. Novak, "Supervised Color Constancy for Machine Vision", *Proceedings of the SPIE: Conference on Visual Processing and Digital Display,* 1991.

[49] C. Novak and S. Shafer, "A Method for Estimating Scene Parameters from Color Histograms", Carnegie Mellon University School of Computer Science, technical report, CMU-CS-93-177, 1993.

[50] Y. Ohta and Y. Hayashi, "Recovery of Illuminant and Surface Colors from Images Based on the CIE Daylight", *Proceedings of the Third European Conference on Computer Vision,* 1994.

[51] B.T. Phong, "Illumination for Computer Generated Images", *Communications of the ACM,* 18:311-317.

[52] J. Piater and P. Utgoff, "Interactively Learning Pixel Classification", submitted to *International Conference on Machine Learning (ICML'97).*

[53] T. Poggio, and F. Girosi, "Regularization Algorithms for Learning that are Equivalent to Multilayer Networks", *Science,* 247:978-982, 1990.

[54] D.A. Pomerleau, Neural Network Perception for Mobile Robot Guidance, Kluwer Academic Publishers, Boston, 1993.

[55] J.R. Quinlan, "Induction of Decision Trees", *Machine Learning,* 1:81-106, 1986.

[56] G. Reynolds, J. R. Beveridge, "Searching for Geometric Structure in Images of Natural Scenes", Proc. of the DARPA Image Understanding Workshop, Los Angeles, CA, February 1987, pp. 257-271. Also, COINS TR 87-03, University of Massachusetts at Amherst, January 1987.

[57] D.E. Rumelhart, G.E. Hinton and J.L. McLelland, "A General Framework for Parallel Distributed Processing", Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Bradford Books/ MIT Press, Cambridge, MA 1986.

[58] Y. Sato, and K. Ikeuchi, "Reflectance Analysis Under Solar Illumination", Proceedings of the *IEEE Workshop for Physics-Based Modeling in Computer Vision,* 1995.

[59] H. Schultz, "Terrain Reconstruction from Widely Separated Images", *Proc. SPIE,* Volume 2486, pp. 113-123, Orlando, FL, April, 1995.

[60] H. Schultz,, "Terrain Reconstruction from Oblique Views", *Proc. ARPA Image Understanding Workshop,* Monterey, CA, November 1994, pp. 1001-1008.

[61] S.A. Shafer, "Using Color to Separate Reflection Components", *Color Research Application,* 10:210-218, 1985.

[62] J.L. Simonds, "Application of Characteristic Vector Analysis to Photographic and Optical Response Data", *Journal of the Optical Society of American,* 53(8), 1963.

[63] G. Tesauro, "Temporal Difference Learning and TD-Gammon", *Communications of the ACM,* 38(3):58-68.

[64] S. Ullman, "Visual Routines", *Cognition,* 18:97-106, 1984. Also in Readings in Computer Vision, Fischler and Firschein (eds.), Morgan-Kaufmann: Los Altos, CA 1987, pp. 371-381.

[65] P. Utgoff, An improved algorithm for incremental induction of decision trees. In Machine Learning: *Proceedings of the Eleventh International Conference,* pages 318--325, New Brunswick, NJ, 1994. Morgan Kaufmann.

[66] P. Utgoff, "Decision tree induction based on efficient tree restructuring", Computer Science Technical Report 95-18, University of Massachusetts, Amherst,MA, 1995.

[67] M.J. Vrhel and H.J. Trussell, "Filter Considerations in Color Correction" *IEEE Transactions on Image Processing,* 3:147-161, 1994.

[68] A. Yuille, "A Method for Computing Spectral Reflectance", *Biological Cybernetics,* 56:195-201, 1987.

[69] W. Zhang and T. Dietterich, "A Reinforcement Learning Approach to Job-Shop Scheduling", *Int. Joint Conference on Artificial Intelligence,* 1995.

[70] X. Wang, Stolle, F., Schultz, H., Riseman, E., Hanson, A., "Using Three-Dimensional Features to Improve Terrain Classification", Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 1997, pp. 915-920.

[71] J. du Buf, M. Kardan, and M. Spann, "Texture feature performance for image segmentation", *Pattern Recognition* 23(3/4), 291--309, 1990.

[72] R. Conners and C. Harlow, "A theoretical comparison of texture algorithms", *IEEE Trans. Pattern Anal. Machine Intell.* 2(3), 204--222, 1980.

[73] D. Foley and J. J. Sammon, "An optimal set of discriminant vectors", *IEEE Trans. on Computers* 24(3), 281--289, 1975.

[74] R., K. Haralick, Shanmugam, and I. Dinstein, "Textural features for image classification", *IEEE Trans. Systems, Man, and Cybernetics* 3(6), 610--621, 1973.

[75] K. Liu, Y. Cheng, and J. Yang, "Algebraic feature extraction for image recognition based on an optimal discriminant criterion", *Pattern Recognition* 26(6), 903--911, 1993.

[76] P. Ohanian and R. Dubes, "Performance evaluation for four classes of textural features", *Pattern Recognition* 25(8), 819--833, 1992.

[77] J. Weszka, C. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification", *IEEE Trans. Systems, Man, and Cybernetics* 6(4), 269--285, 1976.

*U.S. GOVERNMENT PRINTING OFFICE: 1998-610-130-61179

DISTRIBUTION LIST

addresses	number of copies
AFRL/IFEC ATTN: PETER COSTIANES 32 HANGAR ROAD ROME NY 13441-4114	5
UNIVERSITY OF MASSACHUSETTS COMPUTER SCIENCE DEPARTMENT BOX 34610 - LEDERLE GRC BLDG AMHERST MA 01003-4610	5
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
RELIABILITY ANALYSIS CENTER 201 MILL ST. ROME NY 13440-8200	1
ROME LABORATORY/C3AB 525 BROOKS RD ROME NY 13441-4505	1
ATTN: GWEN NGUYEN GIDEP P.O. BOX 8000 CORONA CA 91718-8000	1

AFIT ACADEMIC LIBRARY/LDEE
2950 P STREET
AREA B, BLDG 642
WRIGHT-PATTERSON AFB OH 45433-7765 1

ATTN: R.L. DENISON
WRIGHT LABORATORY/MLPO, BLDG. 651
3005 P STREET, STE 6
WRIGHT-PATTERSON AFB OH 45433-7707 1

ATTN: GILBERT G. KUPERMAN
AL/CFHI, BLDG. 248
2255 H STREET
WRIGHT-PATTERSON AFB OH 45433-7022 1

ATTN: TECHNICAL DOCUMENTS CENTER
OL AL HSC/HRG
2698 G STREET
WRIGHT-PATTERSON AFB OH 45433-7604 1

AIR UNIVERSITY LIBRARY (AUL/LSAD)
600 CHENNAULT CIRCLE
MAXWELL AFB AL 36112-6424 1

US ARMY SSDC
P.O. BOX 1500
ATTN: CSSD-IM-PA
HUNTSVILLE AL 35807-3801 1

NAVAL AIR WARFARE CENTER
WEAPONS DIVISION
CODE 4BL000D
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100 1

SPACE & NAVAL WARFARE SYSTEMS CMD
ATTN: PMW163-1 (R. SKIANO)RM 1044A
53560 HULL ST.
SAN DIEGO, CA 92152-5002 2

SPACE & NAVAL WARFARE SYSTEMS
COMMAND, EXECUTIVE DIRECTOR (PD13A)
ATTN: MR. CARL ANDRIANI
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200 1

COMMANDER, SPACE & NAVAL WARFARE
SYSTEMS COMMAND (CODE 32)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200 — 1

CDR, US ARMY MISSILE COMMAND
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSMI-RD-CS-R, DOCS
REDSTONE ARSENAL AL 35898-5241 — 2

ADVISORY GROUP ON ELECTRON DEVICES
SUITE 500
1745 JEFFERSON DAVIS HIGHWAY
ARLINGTON VA 22202 — 1

REPORT COLLECTION, CIC-14
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545 — 1

AEDC LIBRARY
TECHNICAL REPORTS FILE
100 KINDEL DRIVE, SUITE C211
ARNOLD AFB TN 37389-3211 — 1

COMMANDER
USAISC
ASHC-IMD-L, BLDG 61801
FT HUACHUCA AZ 85613-5000 — 1

US DEPT OF TRANSPORTATION LIBRARY
FB10A, M-457, RM 930
800 INDEPENDENCE AVE, SW
WASH DC 22591 — 1

AWS TECHNICAL LIBRARY
859 BUCHANAN STREET, RM. 427
SCOTT AFB IL 62225-5118 — 1

AFIWC/MSY
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016 — 1

SOFTWARE ENGINEERING INSTITUTE 1
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVENUE
PITTSBURGH PA 15213

NSA/CSS 1
K1
FT MEADE MD 20755-6000

ATTN: OM CHAUHAN 1
DCMC WICHITA
271 WEST THIRD STREET NORTH
SUITE 6000
WICHITA KS 67202-1212

PHILLIPS LABORATORY 1
PL/TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004

ATTN: EILEEN LADUKE/D460 1
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

OUSD(P)/DTSA/DUTD 2
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202